# FC7240 Reference Manual

Rev.0.2

Flagchip

Copyright © 2024 Suzhou Flagchip Semiconductor Co., Ltd. All rights reserved.

This document is the property of the Suzhou Flagchip Semiconductor Co., Ltd. (hereinafter referred to as "Flagchip"). Any information provided in this document is only for reference purposes.

Except as expressly provided herein, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Flagchip. Flagchip hereby disclaims any responsibilities for any losses, expenses, claims, damages, and liabilities related to the use of the information. All information described herein is subject to change without further notice.

ARM and Cortex are registered trademarks of ARM Limited. All other trademarks mentioned in this document are the property of their respective owners.

# Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023/11/20 | Preliminary version |
| 0.2 | 2024/01/15 | • Reorganized *Chapter 2 Features* into three subsections;<br>• Updated *Table 19. WKU input channel connection* in *Section 6.10 Wake-up Unit (WKU);*<br>• Updated clock sources of clock index 6 and 21 in *Table 26. FREQM input sources* in *Section 7.3.7 Frequency Measurement (FREQM);*<br>• Reorganized the general flash information in *Section 8.1 Flash Memory;*<br>• Added a note for booting from ROM to *Section 8.2 SRAM Memory;*<br>• Fixed typos by changing the limit of the PLL VCO clock from "300 MHz to 800 MHz" to "240 MHz to 640 MHz" and the PLL reference clock range from "8 – 48MHz" to "16 – 48 MHz" in *Section 9.1 System Clock Generator (SCG);*<br>• Reorganized the subsections in *Section 9.2 Peripheral Clocks;*<br>• Updated the Failure Description for No.23 DMA0_ERR in *Table 40. FCSMU channel assignments* in *Section 14.2.1 FCSMU Channel Assignments;*<br>• Updated *Table 43. EIM matrix error check (bank channels)* in *Section 14.2.3.3 EIM Channel Error Injection and Check;*<br>• Updated the channel-related descriptions in *Section 14.2.8 ISM;*<br>• Updated some descriptions or figures in chapters: AHB_Overlay, CMU, CPM, CSC, DMA, EIM, FLEXCAN, FREQM, FTU, INTM, MSC, PTIMER, RGM, SCG, SCM, SEC, SENT, SMC, STCU, TMU, TPU, and WKU;<br>• Added an attachment *FC7240 Clocking Diagram.pdf;*<br>• Editorial changes |

Flagchip

# Table of Contents

**Revision History** ..... 3

**Chapter 1 About This Document** ..... 69

1.1 Audience ..... 69

1.2 Abbreviations ..... 69

**Chapter 2 Features** ..... 74

2.1 Block Diagram ..... 74

2.2 Feature Summary ..... 74

2.3 System Diagram ..... 79

**Chapter 3 Cortex-M7 Configuration** ..... 80

**Chapter 4 Memory Map** ..... 81

4.1 System Memory Map ..... 81

4.2 APB Bridge Assignments ..... 82

4.2.1 AFCB0 Slot Assignments ..... 82

4.2.2 AFCB1 Slot Assignments ..... 86

4.3 AHB Peripheral Assignments ..... 91

4.4 PPB Memory Map ..... 91

4.5 Interrupt Vector Assignments ..... 92

4.5.1 WIC Wake-up Sources ..... 98

4.6 DMAMUX Assignments ..... 98

**Chapter 5 Debug and Trace** ..... 103

5.1 Debug Pin Descriptions ..... 105

5.2 JTAG TAP FSM ..... 105

5.3 TAP Connection ..... 107

5.4 ARM AP and DP Connection ..... 107

5.4.1 JTAG and SWD Switching Sequence ..... 108

5.4.2 JTAG DP and AP Connection ..... 108

5.5 SYS AP ..... 110

Flagchip

**Chapter 6 System Block Overview ..... 114**
6.1 Direct Memory Access (DMA) ..... 114
6.2 CPU System Control (CSC) Module ..... 114
6.3 System Control Module (SCM) ..... 114
6.4 CPU Private Module (CPM) ..... 114
6.5 AHB Overlay ..... 114
6.6 Mailbox (MB) ..... 114
6.7 Matrix Access Monitor (MAM) ..... 115
6.8 CORDIC ..... 115
6.9 Security Controller (SEC) ..... 115
6.10 Wake-up Unit (WKU) ..... 115
6.11 Flash Memory Controller (FMC) ..... 116
6.12 Flash Control ..... 116
6.13 On-Chip Connection ..... 116
6.13.1 Logic Unit (LU) ..... 116
6.13.2 Trigger Select (TRGSEL) Sources ..... 116
**Chapter 7 Peripheral Overview ..... 132**
7.1 Human-Machine Interface (HMI) ..... 132
7.1.1 General-Purpose Input/Output (GPIO) ..... 132
7.1.2 Port Control (PORT) ..... 132
7.2 Communication Interface ..... 132
7.2.1 FLEX Controller Area Network (FLEXCAN) ..... 132
7.2.2 FC Serial Peripheral Interface (FCSPI) ..... 132
7.2.3 FC Inter-Integrated Circuit (FCIIC) ..... 133
7.2.4 FC Universal Asynchronous Receiver Transmitter (FCUART) ..... 133
7.2.5 Micro Second Channel (MSC) ..... 133
7.2.6 Single Edge Nibble Transmission (SENT) ..... 133
7.3 Timers ..... 134
7.3.1 Flexible Timer Unit (FTU) ..... 134
7.3.2 FC Programmable Interrupt Timer (FCPIT) ..... 134

Flagchip

7.3.3 Always-on Timer (AONTIMER) ..... 134
7.3.4 Timer Stamp (TSTMP) ..... 134
7.3.5 Programmable Timer (PTIMER) ..... 134
7.3.6 Real-Time Clock (RTC) ..... 134
7.3.7 Frequency Measurement (FREQM) ..... 134
7.4 Analog ..... 136
7.4.1 Analog-to-Digital Converter (ADC) ..... 136
7.4.2 Comparator (CMP) ..... 136
7.4.3 Internal Analog Channels Connection ..... 136
**Chapter 8 Embedded Memory ..... 137**
8.1 Flash Memory ..... 137
8.1.1 Flash Swap Function ..... 137
8.1.2 Flash Configuration Mapping ..... 137
8.1.3 Flash ECC ..... 137
8.1.4 Flash Memory Control ..... 137
8.2 SRAM Memory ..... 138
8.3 ROM ..... 138
**Chapter 9 Clock Overview ..... 139**
9.1 System Clock Generator (SCG) ..... 139
9.2 Peripheral Clocks ..... 141
9.2.1 PCC Assignments ..... 141
9.2.2 Clock Structure ..... 149
**Chapter 10 Reset and Boot Overview ..... 155**
10.1 Reset/Reset Generation Module (RGM) ..... 155
10.2 Boot ..... 156
**Chapter 11 Power Architecture ..... 159**
11.1 Power Connection Diagram ..... 159
11.2 Power Mode ..... 160
11.3 Power Domain ..... 160
11.4 Power Management Controller (PMC)/System Mode Controller (SMC) ..... 160

Flagchip

11.4.1 Exteranl SBC/PMIC Interface ..... 160
**Chapter 12 Security ..... 162**
12.1 NVR Control (Debug etc.) ..... 162
12.2 HSM Subsystem Hardware ..... 172
**Chapter 13 FlexCore Mode ..... 173**
13.1 Instruction Sets Supported by FlexCore ..... 173
13.2 FlexCore Memory ..... 173
13.3 FlexCore AFCB ..... 174
13.4 FlexCore Interrupt ..... 175
13.5 FlexCore SRAM ..... 176
13.6 FlexCore Control ..... 176
13.7 TPU-related Control ..... 176
**Chapter 14 Function Safety Overview ..... 177**
14.1 Introduction ..... 177
14.2 Safety IP Overview ..... 180
14.2.1 FCSMU Channel Assignments ..... 180
14.2.2 STCU ..... 181
14.2.3 EIM/ERM ..... 181
14.2.4 Lockstep Control ..... 195
14.2.5 INTM ..... 195
14.2.6 CMU ..... 196
14.2.7 WDOG ..... 196
14.2.8 ISM ..... 196
14.2.9 TMU ..... 196
14.2.10 CRC ..... 196
**Chapter 15 Pinouts and Packaging ..... 197**
15.1 Pinouts ..... 197
15.2 Package ..... 197
**Chapter 16 CPU Private Module (CPM) ..... 198**
16.1 Revision History ..... 198

Flagchip

16.2 Features ..... 199
16.3 Overview ..... 199
16.4 Functional Description ..... 199
16.4.1 FPU Exception Report ..... 199
16.4.2 Access Priority Control ..... 199
16.4.3 TCM Retry Control ..... 199
16.5 Register Definition ..... 200
16.5.1 CPM_FISCR (FPU Interrupt Status and Control Register) ..... 200
16.5.2 CPM_MISCR (Miscellaneous Control Register) ..... 202
16.5.3 CPM_TCMRCR (TCM Retry Control Register) ..... 203
**Chapter 17 Direct Memory Access (DMA) ..... 206**
17.1 Revision History ..... 206
17.2 Chip-specific Information ..... 207
17.3 Features ..... 207
17.4 Overview ..... 207
17.5 Functional Description ..... 208
17.5.1 DMA Initialization ..... 208
17.5.2 Channel Programming Errors ..... 210
17.5.3 Channel Arbitration Mode ..... 210
17.5.4 DMA Transfer Operations ..... 210
17.5.5 Address Modulo Feature ..... 212
17.5.6 Address Un-align Modulo Feature ..... 213
17.5.7 DMA_CFG ECC Gasket ..... 213
17.5.8 DMA Lockstep Monitor ..... 213
17.5.9 DMAMUX ..... 213
17.6 DMA Register Definition ..... 217
17.6.1 DMA_CR (Control Register) ..... 219
17.6.2 DMA_ES (Error Status Register) ..... 221
17.6.3 DMA_ERQ (Enable Request Register) ..... 222
17.6.4 DMA_EEI (Enable Error Interrupt Register) ..... 224

Flagchip

17.6.5 DMA_CEEI (Clear Enable Error Interrupt Register) ..... 226
17.6.6 DMA_SEEI (Set Enable Error Interrupt Register) ..... 227
17.6.7 DMA_CERQ (Clear Enable Request Register) ..... 227
17.6.8 DMA_SERQ (Set Enable Request Register) ..... 228
17.6.9 DMA_CDNE (Clear DONE Status Bit Register) ..... 228
17.6.10 DMA_SSRT (Set START Bit Register) ..... 229
17.6.11 DMA_CERR (Clear Error Register) ..... 229
17.6.12 DMA_CINT (Clear Interrupt Request Register) ..... 230
17.6.13 DMA_INT (Interrupt Request Register) ..... 230
17.6.14 DMA_ERR (Error Register) ..... 232
17.6.15 DMA_HRS (Hardware Request Status Register) ..... 234
17.6.16 DMA_DUME0 (Un-align Modulo Enable Register 0) ..... 237
17.6.17 DMA_DUMOn (Unalign Modulo N Offset Register) ..... 240
17.6.18 DMA_DCHPRI3 (Channel Priority Register) ..... 241
17.6.19 DMA_DCHPRI2 (Channel Priority Register) ..... 241
17.6.20 DMA_DCHPRI1 (Channel Priority Register) ..... 242
17.6.21 DMA_DCHPRI0 (Channel Priority Register) ..... 242
17.6.22 DMA_DCHPRI7 (Channel Priority Register) ..... 242
17.6.23 DMA_DCHPRI6 (Channel Priority Register) ..... 243
17.6.24 DMA_DCHPRI5 (Channel Priority Register) ..... 243
17.6.25 DMA_DCHPRI4 (Channel Priority Register) ..... 244
17.6.26 DMA_DCHPRI11 (Channel Priority Register) ..... 244
17.6.27 DMA_DCHPRI10 (Channel Priority Register) ..... 245
17.6.28 DMA_DCHPRI9 (Channel Priority Register) ..... 245
17.6.29 DMA_DCHPRI8 (Channel Priority Register) ..... 245
17.6.30 DMA_DCHPRI15 (Channel Priority Register) ..... 246
17.6.31 DMA_DCHPRI14 (Channel Priority Register) ..... 246
17.6.32 DMA_DCHPRI13 (Channel Priority Register) ..... 247
17.6.33 DMA_DCHPRI12 (Channel Priority Register) ..... 247
17.6.34 DMA_CFG_SADDRn (CFG Source Address Register) ..... 248

Flagchip

17.6.35 DMA_CFG_SOFFn (CFG Signed Source Address Offset Register) ..... 248
17.6.36 DMA_CFG_ATTRn (CFG Transfer Attributes Register) ..... 248
17.6.37 DMA_CFG_NBYTES_ILNOn (CFG Inner Byte Count Inner Loop Mapping Disabled Register) ..... 249
17.6.38 DMA_CFG_NBYTES_ILOFFNOn (CFG Signed Inner Loop Offset Inner Loop Mapping Enabled and Offset Disabled Register) ..... 250
17.6.39 DMA_CFG_NBYTES_ILOFFYESn (CFG Signed Inner Loop Offset Inner Loop Mapping and Offset Enabled Register) ..... 251
17.6.40 DMA_CFG_SLASTn (CFG Last Source Address Adjustment Register) ..... 252
17.6.41 DMA_CFG_DADDRn (CFG Destination Address Register) ..... 253
17.6.42 DMA_CFG_DOFFn (CFG Signed Destination Address Offset Register) ..... 253
17.6.43 DMA_CFG_CLC_CHTRGENNOn (CFG Current Inner Loop Trigger, Outer Loop Count Channel Trigger Disabled Register) ..... 254
17.6.44 DMA_CFG_CLC_CHTRGENYESn (CFG Current Inner Loop Trigger, Outer Loop Count Channel Trigger Enabled Register) ..... 255
17.6.45 DMA_CFG_DLASTn (CFG Last Destination Address Adjustment Register) ..... 256
17.6.46 DMA_CFG_CSRn (CFG Control and Status Register) ..... 257
17.6.47 DMA_CFG_BLC_CHTRGENNOn (CFG Beginning Inner Loop Trigger, Outer Loop Count Channel Trigger Disabled Register) ..... 259
17.6.48 DMA_CFG_BLC_CHTRGENYESn (CFG Beginning Inner Loop Trigger, Outer Loop Count Channel Trigger Enabled Register) ..... 259
17.7 DMAMUX Register Definition ..... 262
17.7.1 DMAMUX_CHCFGn (Channel N Configuration Register) ..... 262
17.7.2 DMAMUX_CHTRG (Channel N Trigger Register) ..... 262
**Chapter 18 Wake-up Unit (WKU) ..... 264**
18.1 Revision History ..... 264
18.2 Chip-specific Information ..... 265
18.3 Features ..... 265
18.4 Overview ..... 265
18.5 Functional Description ..... 265
18.5.1 WKU Delay Counter ..... 265
18.6 Register Definition ..... 266

Flagchip

18.6.1 WKU_MDC (Module Delay Configuration Register0) ..... 266
18.6.2 WKU_MWER0 (Module Wakeup Enable Register0) ..... 267
18.6.3 WKU_MWER1 (Module Wakeup Enable Register1) ..... 268
18.6.4 WKU_MWER2 (Module Wakeup Enable Register2) ..... 269
**Chapter 19 Power Management Controller (PMC) ..... 271**
19.1 Revision History ..... 271
19.2 Features ..... 272
19.3 Overview ..... 272
19.4 Block Diagram ..... 272
19.5 Functional Description ..... 272
19.5.1 Regulators ..... 272
19.5.2 Resets ..... 273
19.5.3 Interrupts ..... 273
19.6 Register Definition ..... 274
19.6.1 PMC_LVSCR (Low Voltage Status and Control Register) ..... 274
19.6.2 PMC_CONFIG (PMC Configuration Register) ..... 277
**Chapter 20 System Mode Controller (SMC) ..... 279**
20.1 Revision History ..... 279
20.2 Features ..... 280
20.2.1 Introduction ..... 280
20.3 Overview ..... 280
20.3.1 Modes of Operation ..... 280
20.4 Functional Description ..... 281
20.4.1 Power Mode Transitions ..... 281
20.4.2 Low-Power Entry/Exit Sequence ..... 281
20.5 Register Definition ..... 282
20.5.1 SMC_PMCTRL (Power Mode Control Register) ..... 282
20.5.2 SMC_STANDBY_CFG (Standby Mode Configuration Register) ..... 283
**Chapter 21 Reset Generation Module (RGM) ..... 285**
21.1 Revision History ..... 285

Flagchip

21.2 Features ..... 286
21.3 Overview ..... 286
21.4 Functional Description ..... 286
21.4.1 Reset Sources ..... 286
21.4.2 Reset Interrupt ..... 288
21.4.3 Reset Pin Filter ..... 288
21.5 Register Definition ..... 289
21.5.1 RGM_SRS (System Reset Status Register) ..... 289
21.5.2 RGM_RSTFLT (Reset Filter Control Register) ..... 291
21.5.3 RGM_SSRS (Sticky System Reset Status Register) ..... 292
21.5.4 RGM_SRIE (System Reset Interrupt Enable Register) ..... 295
21.5.5 RGM_C0_CFG (CPU0 Reset Configuration Register) ..... 296
21.5.6 RGM_C0_RST (CPU0 Reset Register) ..... 298
21.5.7 RGM_C0_SRS (CPU0 System Reset Status Register) ..... 298
21.5.8 RGM_C0_SSRS (CPU0 Sticky System Reset Status Register) ..... 300
**Chapter 22 Security Controller (SEC) ..... 302**
22.1 Revision History ..... 302
22.2 Features ..... 303
22.3 Overview ..... 303
22.4 Functional Description ..... 303
22.4.1 NVR Load ..... 303
22.4.2 Registers Write Lock ..... 303
22.4.3 Debug Re-Enable Mechanism ..... 303
22.4.4 Test Mode Re-enable Mechanism ..... 303
22.4.5 Mass Erase Disable ..... 303
22.4.6 Chip Unlock ..... 304
22.4.7 Secure Boot ..... 304
22.4.8 Non-Secure Boot HSM Part ..... 304
22.4.9 Non-Secure Boot Co-processor Part ..... 305
22.4.10 Chip Boot Address ..... 305

Flagchip

22.5 Register Definition ..... 305
22.5.1 SEC_DEN (Debug Enable Register) ..... 306
22.5.2 SEC_FSEC0 (System Security_Control Register0) ..... 307
22.5.3 SEC_FSEC1 (System Security_Control Register1) ..... 307
22.5.4 SEC_DCWOR (Debug Control Write Once Register) ..... 308
22.5.5 SEC_DEKn (Debug Re-Enable Key Register) ..... 309
22.5.6 SEC_TME (Test Mode Enable Register) ..... 309
22.5.7 SEC_TMEA (Test Mode Re-Enable Allow Register) ..... 310
22.5.8 SEC_TMEK (Test Mode Re-Enable Key Register) ..... 311
22.5.9 SEC_FCR0 (Flash Control Register0) ..... 311
22.5.10 SEC_NKRP (NVR Key Read Protection) ..... 312
22.5.11 SEC_BCS (Boot Configuration Status Register) ..... 312
22.5.12 SEC_UKAC (User Key Access Configuration Register) ..... 313
22.5.13 SEC_BRC0 (BootROM Configuration Register0) ..... 314
22.5.14 SEC_BRC1 (BootROM Configuration Register1) ..... 315
22.5.15 SEC_SW_CFG_ISP_FLG (Software Configuration to Enter ISP Flag Register) ..... 317
22.5.16 SEC_BRC2 (BootROM Configuration Register2) ..... 317
22.5.17 SEC_IMGEA (Image Address Register) ..... 318
22.5.18 SEC_NVR_VER (NVR Version Register) ..... 319
22.5.19 SEC_LCSTAT (Lifecycle Status Register) ..... 320
22.5.20 SEC_FAC (Flash Access Control Register) ..... 321
22.5.21 SEC_FLEXCORE_EN (FlexCore Enable Register) ..... 322
22.5.22 SEC_FLEX_CODE_ADDR (Code Address for FlexCore Register) ..... 323
22.5.23 SEC_PFLASH_PRLLL_EN (PFLASH Parallel Enable Register) ..... 323
**Chapter 23 CORDIC ..... 325**
23.1 Revision History ..... 325
23.2 Chip-specific Information ..... 326
23.3 Features ..... 326
23.4 Overview ..... 326
23.5 Block Diagram ..... 327

Flagchip

23.6 Functional Description ..... 327
23.6.1 Numerical Representation ..... 327
23.6.2 CORDIC Function Summary ..... 327
23.6.3 CORDIC Extended Functions ..... 331
23.7 Register Definition ..... 332
23.7.1 CORDIC_CTRL (Control Register) ..... 332
23.7.2 CORDIC_X_INPUT (X-axis Input Data Register) ..... 333
23.7.3 CORDIC_Y_INPUT (Y-axis Input Data Register) ..... 334
23.7.4 CORDIC_Z_INPUT (Z-axis Input Data Register) ..... 334
23.7.5 CORDIC_STAT (State Register) ..... 335
23.7.6 CORDIC_X_OUTPUT (X-axis Output Data Register) ..... 335
23.7.7 CORDIC_Y_OUTPUT (Y-axis Output Data Register) ..... 336
23.7.8 CORDIC_Z_OUTPUT (Z-axis Output Data Register) ..... 336
**Chapter 24 CPU System Control (CSC) ..... 338**
24.1 Revision History ..... 338
24.2 Chip-specific Information ..... 339
24.3 Features ..... 339
24.4 Overview ..... 339
24.5 Functional Description ..... 339
24.5.1 Peripherals Low Power Handshake ..... 339
24.5.2 CSC Register LOCK Function ..... 339
24.5.3 CSC Register WPB_LOCK and WPB Function ..... 339
24.6 CSC0 Register Definition ..... 340
24.6.1 CSC0_STOP_MODER0 (STOP Mode Control Register 0) ..... 340
24.6.2 CSC0_STOP_REQR0 (STOP Request Register 0) ..... 342
24.6.3 CSC0_STOP_ACKR0 (STOP Acknowledge Register 0) ..... 344
24.6.4 CSC0_STOP_MODER1 (STOP Mode Control Register 1) ..... 346
24.6.5 CSC0_STOP_REQR1 (STOP Request Register 1) ..... 348
24.6.6 CSC0_STOP_ACKR1 (STOP Acknowledge Register 1) ..... 350
24.6.7 CSC0_STOP_MODER2 (STOP Mode Control Register 2) ..... 351

Flagchip

24.6.8 CSC0_STOP_REQR2 (STOP Request Register 2) ..... 353
24.6.9 CSC0_STOP_ACKR2 (STOP Acknowledge Register 2) ..... 354
24.6.10 CSC0_CCM0_CFG (CCM0 Configuration Register) ..... 355
24.6.11 CSC0_SCG_MAM_STALL (SCG MAM Stall Request Register) ..... 356
24.6.12 CSC0_CPU0_INT (CPU0 Software Interrupt Register) ..... 357
24.6.13 CSC0_CLKOUT_CTRL (Clkout Control Register) ..... 358
24.6.14 CSC0_AONCLKSR (AON Clock Select Register) ..... 359
24.6.15 CSC0_PCU_CTRL (PCU Control Register) ..... 360
24.6.16 CSC0_SMU_CTRL0 (FCSMU Control Register 0) ..... 361
24.6.17 CSC0_SMU_CTRL1 (FCSMU Control Register 1) ..... 365
24.6.18 CSC0_SMU_CTRL4 (FCSMU Control Register 4) ..... 367
24.6.19 CSC0_CMU_CTRL (CMU Control Register) ..... 369
24.6.20 CSC0_SMU_CTRL5 (FCSMU Control Register 5) ..... 370
24.6.21 CSC0_LP_WAKEUP (Low Power Wakeup Register) ..... 372
**Chapter 25 System Control Module (SCM) ..... 377**
25.1 Revision History ..... 377
25.2 Chip-specific Information ..... 378
25.3 Features ..... 378
25.4 Overview ..... 378
25.5 Functional Description ..... 378
25.5.1 SCM CRC List ..... 378
25.5.2 SCM Register LOCK Function ..... 379
25.5.3 SCM Register WPB_LOCK and WPB Function ..... 379
25.6 Register Definition ..... 379
25.6.1 SCM_UIDL (Unique Identification Register 0) ..... 380
25.6.2 SCM_UIDML (Unique Identification Register 1) ..... 381
25.6.3 SCM_UIDMH (Unique Identification Register 2) ..... 381
25.6.4 SCM_UIDH (Unique Identification Register 3) ..... 382
25.6.5 SCM_PARTID0 (PART ID Register) ..... 382
25.6.6 SCM_CHIPCFG0 (CHIPCFG Register 0) ..... 383

Flagchip

25.6.7 SCM_CHIPCFG1 (CHIPCFG Register 1) ..... 384
25.6.8 SCM_MAMECCEN0 (MAM ECC Enable Register 0) ..... 384
25.6.9 SCM_MAMECCEN1 (MAM ECC Enable Register 1) ..... 387
25.6.10 SCM_CPU0ECCEN (CPU0 ECC Enable Register) ..... 389
25.6.11 SCM_SRAM_EDC_CTRL (SRAM EDC Control Register) ..... 391
25.6.12 SCM_ROMCFG (ROM Configuration Register) ..... 391
25.6.13 SCM_FCSPI_ROUTING (FCSPI Routing Register) ..... 392
25.6.14 SCM_FCUART_ROUTING0 (FCUART Routing Register 0) ..... 394
25.6.15 SCM_ADC_ROUTING (ADC Routing Register) ..... 396
25.6.16 SCM_FTU_ROUTING (FTU Routing Register) ..... 399
25.6.17 SCM_FTU_GTBC (FTU Global Time Base Control Register) ..... 400
25.6.18 SCM_FTU_SYNC (FTU SYNC Register) ..... 401
25.6.19 SCM_DEBUG_TRACE (DEBUG TRACE Register) ..... 402
25.6.20 SCM_SOCMISC (SOCMISC Register) ..... 403
25.6.21 SCM_CCM0_STATUS (CCM0 Status Register) ..... 404
25.6.22 SCM_FLEXCAN_ROUTING (FLEXCAN Routing Register) ..... 405
25.6.23 SCM_MSC0_ROUTING (MSC0 Routing Register) ..... 406
25.6.24 SCM_ISM_ROUTING (ISM Routing Register) ..... 408
25.6.25 SCM_MATRIX_STATUS0 (Matrix Status Register 0) ..... 410
25.6.26 SCM_MATRIX_STATUS1 (Matrix Status Register 1) ..... 412
25.6.27 SCM_MATRIX_STATUS2 (Matrix Status Register 2) ..... 414
25.6.28 SCM_MATRIX_STATUS5 (Matrix Status Register 5) ..... 416
25.6.29 SCM_MATRIX_ID_STATUS0 (Matrix Master ID Status Register) ..... 418
25.6.30 SCM_MATRIX_STATUS6 (Matrix Status Register 6) ..... 420
25.6.31 SCM_MATRIX_STATUS7 (Matrix Status Register 7) ..... 422
25.6.32 SCM_TPU_GTBCM (TPU Global Time Base Control Mask Register) ..... 423
25.6.33 SCM_FTU_GTBCM (FTU Global Time Base Control Mask Register) ..... 423
25.6.34 SCM_SYSAP_MDO (SYSAP MDO Register) ..... 425
25.6.35 SCM_SYSAP_MDI (SYSAP MDI Register) ..... 425
25.6.36 SCM_SYSAP_CTRL (SYSAP Control Register) ..... 426

Flagchip

25.6.37 SCM_SUBSYS_PCC (Subsystem PCC Register) ..... 428
25.6.38 SCM_SUBSYS_STATUS (Subsystem Status Register) ..... 429
25.6.39 SCM_MDO_FLAG (Mailbox Data Output Flag Register) ..... 430
25.6.40 SCM_MASTER_HALT_REQ (Master Halt Request Register) ..... 431
25.6.41 SCM_MASTER_HALT_ACK (Master Halt ACK Register) ..... 432
25.6.42 SCM_INT_ROUTER_NMI (NMI Interrupt Router Register) ..... 432
25.6.43 SCM_CRCCSR (CRC Control Status Register) ..... 433
25.6.44 SCM_CRCRES (CRC Result Register) ..... 434
**Chapter 26 Matrix Access Monitor (MAM) ..... 435**
26.1 Revision History ..... 435
26.2 Chip-specific Information ..... 436
26.3 Features ..... 439
26.4 Overview ..... 439
26.5 Functional Description ..... 439
26.5.1 Access Control ..... 440
26.5.2 Block Access Control Policy Lock ..... 440
26.5.3 Watchdog Timer ..... 440
26.6 Register Definition ..... 440
26.6.1 MAM_MAXCFG (Matrix Configuration Register) ..... 441
26.6.2 MAM_BSTCR (Burst Control Register) ..... 441
26.6.3 MAM_WDGCR (Watchdog Control Register) ..... 442
26.6.4 MAM_TOCR (Timeout Control Register) ..... 443
26.6.5 MAM_WDGDIV (Watchdog Divider Register) ..... 444
26.6.6 MAM_ACLRn (Access Control Lock Register) ..... 445
26.6.7 MAM_ACRn (Access Control Register) ..... 448
**Chapter 27 Mailbox (MB) ..... 451**
27.1 Revision History ..... 451
27.2 Chip-specific Information ..... 452
27.3 Features ..... 452
27.4 Overview ..... 452

Flagchip

27.5 Functional Description ..... 453
27.5.1 Acquire Communication Channels ..... 453
27.5.2 Response Error ..... 453
27.5.3 Locks for Mutual Exclusion Access ..... 453
27.5.4 Short Message Passing ..... 453
27.5.5 Long Message Passing ..... 454
27.6 Register Definition ..... 455
27.6.1 MB_CCn_SEMA (Communication Channel Smeaphore Register) ..... 455
27.6.2 MB_CCn_SEMA_UNLK (Communication Channel Smeaphore Unlock Register) ..... 456
27.6.3 MB_CCn_REQUEST (Communication Channel Request Register) ..... 457
27.6.4 MB_CCn_DONE (Communication Channel Done Register) ..... 458
27.6.5 MB_CCn_DONE_MASK (Communication Channel Done Mask Register) ..... 458
27.6.6 MB_CCn_DATA0 (Communication Channel Data Register0) ..... 459
27.6.7 MB_CCn_DATA1 (Communication Channel Data Register1) ..... 459
27.6.8 MB_CCn_STAT (Communication Channel Status Register) ..... 460
27.6.9 MB_CCn_CLR (Communication Channel Clear Register) ..... 461
27.6.10 MB_INTn_FLG (Interrupt Channel Flag Register) ..... 461
27.6.11 MB_INTn_FLG_MASK (Interrupt Channel Flag Mask Register) ..... 462
27.6.12 MB_INTn_INTEN (Interrupt Channel Interrupt Enable Register) ..... 463
27.6.13 MB_INTn_FLG_STAT (Interrupt Channel Flag Status Register) ..... 464
27.6.14 MB_INTn_CTRL (Interrupt Channel Control Register) ..... 464
**Chapter 28 Flash Memory Controller (FMC) ..... 466**
28.1 Revision History ..... 466
28.2 Chip-specific Information ..... 467
28.3 Features ..... 467
28.4 Overview ..... 467
28.5 Functional Description ..... 467
28.5.1 Read Access ..... 468
28.5.2 Write Access ..... 468
28.5.3 Error Response ..... 468

Flagchip

28.5.4 Read Buffers and Prefetch ..... 468
28.5.5 Read Buffer ECC and Error Injection ..... 468
28.5.6 Flash Coarse and Fine Region Protection ..... 469
28.5.7 Flash ECC Error Report ..... 469
28.5.8 Flash Read While Write Hold ..... 469
28.5.9 Flash OTA ..... 469
28.6 Register Definition ..... 470
28.6.1 FMC_FAPC0 (Flash Access Port Control Register 0) ..... 471
28.6.2 FMC_FAPC1 (Flash Access Port Control Register 1) ..... 472
28.6.3 FMC_FAPC2 (Flash Access Port Control Register 2) ..... 473
28.6.4 FMC_FEEC (Flash ECC Error Control Register) ..... 474
28.6.5 FMC_FEIPC (Flash ECC Inject Position Control Register) ..... 475
28.6.6 FMC_FPESA_L (Flash Program Erase Start Address Logical Register) ..... 476
28.6.7 FMC_FPESA_P (Flash Program Erase Start Address Physical Register) ..... 476
28.6.8 FMC_FB_FPELCKn (Flash Block n Fine Program Erase Lock Register) ..... 477
28.6.9 FMC_FN_FPELCK (Flash NVR Fine Program Erase Lock Register) ..... 477
28.6.10 FMC_FB_CPELCKn (Flash Block n Coarse Program Erase Lock Register) ..... 478
28.6.11 FMC_OTA_CTRLn (FLASH OTA Control Register) ..... 478
28.6.12 FMC_OTA_VER_LOCn (FLASH OTA Version Location Register) ..... 479
28.6.13 FMC_OTA_ACT_VERn (FLASH OTA Active Version Register) ..... 480
**Chapter 29 ARMv7 Cache Controller (ACC) ..... 481**
29.1 Revision History ..... 481
29.2 Chip-specific Information ..... 482
29.3 Features ..... 482
29.4 Overview ..... 482
29.5 Functional Description ..... 482
29.5.1 DCACHE Enable ..... 482
29.5.2 ICACEHE Enable ..... 482
29.5.3 DCACHE Disable ..... 483
29.5.4 ICACHE Disable ..... 483

Flagchip

29.6 Register Definition ..... 483
29.6.1 ACC_CCR (Configuration and Control Register) ..... 483
29.6.2 ACC_CLIDR (Cache Level ID Register) ..... 486
29.6.3 ACC_CTR (Cache Type Register) ..... 486
29.6.4 ACC_CCSIDR (Cache Size ID Register) ..... 488
29.6.5 ACC_CSSELR (Cache Size Selection Register) ..... 489
29.6.6 ACC_ICIALLU (I-cache Invalidate All to PoU Register) ..... 489
29.6.7 ACC_ICIMVAU (I-cache Invalidate by MVA to PoU Register) ..... 490
29.6.8 ACC_DCIMVAC (D-cache Invalidate by MVA to PoC Register) ..... 490
29.6.9 ACC_DCISW (D-cache Invalidate by Set-way Register) ..... 491
29.6.10 ACC_DCCMVAU (D-cache Clean by MVA to PoU Register) ..... 492
29.6.11 ACC_DCCMVAC (D-cache Clean by MVA to PoC Register) ..... 492
29.6.12 ACC_DCCSW (D-cache Clean by Set-way Register) ..... 493
29.6.13 ACC_DCCIMVAC (D-cache Clean and Invalidate by MVA to PoC Register) ..... 493
29.6.14 ACC_DCCISW (D-cache Clean and Invalidate by Set-way Register) ..... 494
29.6.15 ACC_CACR (L1 Cache Control Register) ..... 494
29.6.16 ACC_IEBRn (Instruction Error Bank Register) ..... 496
29.6.17 ACC_DEBRn (Data Error Bank Register) ..... 496
**Chapter 30 AHB Overlay ..... 498**
30.1 Revision History ..... 498
30.2 Chip-specific Information ..... 499
30.3 Features ..... 499
30.4 Overview ..... 499
30.5 Functional Description ..... 500
30.5.1 AHB Overlay Remapping Region ..... 500
30.5.2 FAR Region ..... 500
30.6 Register Definition ..... 500
30.6.1 AHB_OVERLAY_REGION_0_SRC (Region 0 Source Address Register) ..... 501
30.6.2 AHB_OVERLAY_REGION_0_DST (Region 0 Destination Address Register) ..... 502
30.6.3 AHB_OVERLAY_REGION_0_SIZE (Region 0 Size Register) ..... 502

Flagchip

30.6.4 AHB_OVERLAY_REGION_0_EN (Region 0 Enable Register) ..... 503
30.6.5 AHB_OVERLAY_REGION_1_SRC (Region 1 Source Address Register) ..... 503
30.6.6 AHB_OVERLAY_REGION_1_DST (Region 1 Destination Address Register) ..... 504
30.6.7 AHB_OVERLAY_REGION_1_SIZE (Region 1 Size Register) ..... 504
30.6.8 AHB_OVERLAY_REGION_1_EN (Region 1 Enable Register) ..... 505
30.6.9 AHB_OVERLAY_REGION_2_SRC (Region 2 Source Address Register) ..... 506
30.6.10 AHB_OVERLAY_REGION_2_DST (Region 2 Destination Address Register) ..... 506
30.6.11 AHB_OVERLAY_REGION_2_SIZE (Region 2 Size Register) ..... 507
30.6.12 AHB_OVERLAY_REGION_2_EN (Region 2 Enable Register) ..... 507
30.6.13 AHB_OVERLAY_GLOBAL_EN (FAR and Overlay Enable Register) ..... 508
30.6.14 AHB_OVERLAY_INTR_FLAG (Interrupt Flag Register) ..... 508
30.6.15 AHB_OVERLAY_INTR_CLR (Interupt Clear Register) ..... 511
30.6.16 AHB_OVERLAY_INTR_EN (Interrupt Enable Register) ..... 513
30.6.17 AHB_OVERLAY_FAR_SRC (FAR Source Register) ..... 513
30.6.18 AHB_OVERLAY_FAR_DST (FAR Destination Register) ..... 514
30.6.19 AHB_OVERLAY_FAR_SIZE (FAR Size Register) ..... 514
30.6.20 AHB_OVERLAY_REGION0_MASK (Region 0 Mask Register) ..... 515
30.6.21 AHB_OVERLAY_REGION1_MASK (Region 1 Mask Register) ..... 516
30.6.22 AHB_OVERLAY_REGION2_MASK (Region 2 Mask Register) ..... 517
**Chapter 31 System Clock Generator (SCG) ..... 519**
31.1 Revision History ..... 519
31.2 Chip-specific Information ..... 520
31.3 Features ..... 520
31.4 Overview ..... 520
31.5 Functional Description ..... 521
31.5.1 Clock Enable in Run Mode ..... 521
31.5.2 Clock Status in Low Power Mode ..... 521
31.5.3 Function Clock Configuration ..... 522
31.5.4 System Clock Switch Sequence ..... 522
31.5.5 FIRC/SIRC Automatic Trim ..... 522

Flagchip

31.5.6 NVM Clock Configuration ..... 522
31.5.7 CMU4 Reference Clock Configuration ..... 523
31.5.8 Internal Watchdog ..... 523
31.5.9 Configuration CRC Check ..... 523
31.6 Register Definition ..... 524
31.6.1 SCG_CSR (Clock Status Register) ..... 525
31.6.2 SCG_CCR (Clock Control Register) ..... 526
31.6.3 SCG_CLKOUTCFG (CLKOUT Configuration Register) ..... 527
31.6.4 SCG_SLPWDG (Low Power Sleep WDOG Register) ..... 528
31.6.5 SCG_WKPWDG (Low Power Wakeup WDOG Register) ..... 529
31.6.6 SCG_CRCCSR (CRC Control Status Register) ..... 529
31.6.7 SCG_CRCRES (CRC Result Register) ..... 530
31.6.8 SCG_PLL_LOLC (PLL Loss of Lock Control Register) ..... 531
31.6.9 SCG_FOSCCSR (Fast OSC Control Status Register) ..... 531
31.6.10 SCG_FOSCDIV (Fast OSC Divide Register) ..... 533
31.6.11 SCG_FOSCCFG (Fast OSC Configuration Register) ..... 534
31.6.12 SCG_SIRCCSR (Slow IRC Control Status Register) ..... 535
31.6.13 SCG_SIRCDIV (Slow IRC Divide Register) ..... 537
31.6.14 SCG_SIRCTCFG (Slow IRC Auto Trim Configuration Register) ..... 538
31.6.15 SCG_SIRCSTAT (Slow IRC Software Trim Status Register) ..... 539
31.6.16 SCG_FIRCCSR (Fast IRC Control Status Register) ..... 540
31.6.17 SCG_FIRCDIV (Fast IRC Divide Register) ..... 541
31.6.18 SCG_FIRCCFG (Fast IRC Configuration Register) ..... 543
31.6.19 SCG_FIRCTCFG (Fast IRC Auto Trim Configuration Register) ..... 544
31.6.20 SCG_FIRCSTAT (Fast IRC Software Trim Status Register) ..... 545
31.6.21 SCG_SOSCCSR (Slow OSC Control Status Register) ..... 546
31.6.22 SCG_SOSCCFG (Slow OSC Configuration Register) ..... 547
31.6.23 SCG_PLL1CSR (PLL1 Control Status Register) ..... 548
31.6.24 SCG_PLL1DIV (PLL1 Divide Register) ..... 549
31.6.25 SCG_PLL1CFG (PLL1 Configuration Register) ..... 551

Flagchip

31.6.26 SCG_PLL0CSR (PLL0 Control Status Register) ..... 552
31.6.27 SCG_PLL0DIV (PLL0 Divide Register) ..... 553
31.6.28 SCG_PLL0CFG (PLL0 Configuration Register) ..... 555
31.6.29 SCG_SIRC32KCSR (32K Slow IRC Control Status Register) ..... 556
**Chapter 32 Peripheral Configuration Control (PCC) ..... 558**
32.1 Revision History ..... 558
32.2 Chip-specific Information ..... 559
32.3 Features ..... 559
32.4 Overview ..... 559
32.4.1 Peripheral Clock Control ..... 559
32.4.2 Peripheral Reset Control ..... 560
32.5 Functional Description ..... 560
32.6 Register Definition ..... 560
32.6.1 PCC_DMA0 (DMA0 Clock Control Register) ..... 562
32.6.2 PCC_DMAMUX0 (DMAMUX0 Clock Control Register) ..... 563
32.6.3 PCC_ROMC (ROMC Clock Control Register) ..... 563
32.6.4 PCC_ERM (ERM Clock Control Register) ..... 564
32.6.5 PCC_EIM (EIM Clock Control Register) ..... 565
32.6.6 PCC_INTM0 (INTM0 Clock Control Register) ..... 565
32.6.7 PCC_ISM0 (ISM0 Clock Control Register) ..... 566
32.6.8 PCC_WDOG0 (WDOG0 Clock Control Register) ..... 567
32.6.9 PCC_TRGSEL0 (TRGSEL0 Clock Control Register) ..... 567
32.6.10 PCC_TRGSEL1 (TRGSEL1 Clock Control Register) ..... 568
32.6.11 PCC_TRGSEL2 (TRGSEL2 Clock Control Register) ..... 568
32.6.12 PCC_TRGSEL3 (TRGSEL3 Clock Control Register) ..... 569
32.6.13 PCC_CRC0 (CRC0 Clock Control Register) ..... 570
32.6.14 PCC_CORDIC0 (CORDIC0 Clock Control Register) ..... 570
32.6.15 PCC_TSTMP0 (TSTMP0 Clock Control Register) ..... 571
32.6.16 PCC_TSTMP1 (TSTMP1 Clock Control Register) ..... 572
32.6.17 PCC_FCPIT0 (FCPIT0 Clock Control Register) ..... 572

Flagchip

32.6.18 PCC_AONTIMER0 (AONTIMER0 Clock Control Register) ..... 573
32.6.19 PCC_RTC (RTC Clock Control Register) ..... 574
32.6.20 PCC_CMU0 (CMU0 Clock Control Register) ..... 575
32.6.21 PCC_CMU1 (CMU1 Clock Control Register) ..... 575
32.6.22 PCC_CMU2 (CMU2 Clock Control Register) ..... 576
32.6.23 PCC_CMU3 (CMU3 Clock Control Register) ..... 577
32.6.24 PCC_CMU4 (CMU4 Clock Control Register) ..... 577
32.6.25 PCC_PTIMER0 (PTIMER0 Clock Control Register) ..... 578
32.6.26 PCC_PTIMER1 (PTIMER1 Clock Control Register) ..... 579
32.6.27 PCC_ADC0 (ADC0 Clock Control Register) ..... 579
32.6.28 PCC_ADC1 (ADC1 Clock Control Register) ..... 581
32.6.29 PCC_WKU (WKU Clock Control Register) ..... 582
32.6.30 PCC_CMP0 (CMP0 Clock Control Register) ..... 582
32.6.31 PCC_CMP1 (CMP1 Clock Control Register) ..... 583
32.6.32 PCC_TMU (TMU Clock Control Register) ..... 584
32.6.33 PCC_SENT0 (SENT0 Clock Control Register) ..... 585
32.6.34 PCC_MB (MB Clock Control Register) ..... 586
32.6.35 PCC_FTU0 (FTU0 Clock Control Register) ..... 586
32.6.36 PCC_FTU1 (FTU1 Clock Control Register) ..... 587
32.6.37 PCC_FTU2 (FTU2 Clock Control Register) ..... 588
32.6.38 PCC_FTU3 (FTU3 Clock Control Register) ..... 589
32.6.39 PCC_FCSPI0 (FCSPI0 Clock Control Register) ..... 590
32.6.40 PCC_FCSPI1 (FCSPI1 Clock Control Register) ..... 591
32.6.41 PCC_FCSPI2 (FCSPI2 Clock Control Register) ..... 592
32.6.42 PCC_FCIIC0 (FCIIC0 Clock Control Register) ..... 593
32.6.43 PCC_FCUART0 (FCUART0 Clock Control Register) ..... 594
32.6.44 PCC_FCUART1 (FCUART1 Clock Control Register) ..... 595
32.6.45 PCC_FCUART2 (FCUART2 Clock Control Register) ..... 596
32.6.46 PCC_FCUART3 (FCUART3 Clock Control Register) ..... 597
32.6.47 PCC_LU0 (LU0 Clock Control Register) ..... 598

Flagchip

32.6.48 PCC_FREQM (FREQM Clock Control Register) ..... 599
32.6.49 PCC_STCU (STCU Clock Control Register) ..... 599
32.6.50 PCC_FLEXCAN0 (FLEXCAN0 Clock Control Register) ..... 600
32.6.51 PCC_FLEXCAN1 (FLEXCAN1 Clock Control Register) ..... 601
32.6.52 PCC_WDOG1 (WDOG1 Clock Control Register) ..... 602
32.6.53 PCC_TRGSEL4 (TRGSEL4 Clock Control Register) ..... 603
32.6.54 PCC_TRGSEL5 (TRGSEL5 Clock Control Register) ..... 603
32.6.55 PCC_FCSPI3 (FCSPI3 Clock Control Register) ..... 604
32.6.56 PCC_FCSPI4 (FCSPI4 Clock Control Register) ..... 605
32.6.57 PCC_FCSPI5 (FCSPI5 Clock Control Register) ..... 606
32.6.58 PCC_FTU4 (FTU4 Clock Control Register) ..... 607
32.6.59 PCC_FTU5 (FTU5 Clock Control Register) ..... 608
32.6.60 PCC_FTU6 (FTU6 Clock Control Register) ..... 609
32.6.61 PCC_FTU7 (FTU7 Clock Control Register) ..... 610
32.6.62 PCC_FCIIC1 (FCIIC1 Clock Control Register) ..... 611
32.6.63 PCC_FCUART4 (FCUART4 Clock Control Register) ..... 612
32.6.64 PCC_FCUART5 (FCUART5 Clock Control Register) ..... 613
32.6.65 PCC_FCUART6 (FCUART6 Clock Control Register) ..... 614
32.6.66 PCC_FCUART7 (FCUART7 Clock Control Register) ..... 615
32.6.67 PCC_MSC0 (MSC0 Clock Control Register) ..... 616
32.6.68 PCC_FLEXCAN2 (FLEXCAN2 Clock Control Register) ..... 617
32.6.69 PCC_FLEXCAN3 (FLEXCAN3 Clock Control Register) ..... 618
**Chapter 33 Watchdog (WDOG) ..... 620**
33.1 Revision History ..... 620
33.2 Chip-specific Information ..... 621
33.2.1 Clock Configuration ..... 621
33.2.2 Watchdog Timeout and Reaction ..... 621
33.3 Overview ..... 621
33.4 Features ..... 621
33.5 Block Diagram ..... 622

Flagchip

33.6 Functional Description ..... 622
33.6.1 Configuration of Watchdog ..... 622
33.6.2 Watchdog Refresh Mechanism ..... 623
33.6.3 Interrupt Reaction Mode ..... 623
33.6.4 Backup Reset ..... 623
33.6.5 Self-check Mechanism of Watchdog ..... 624
33.6.6 Clock Source ..... 624
33.7 Application Information ..... 625
33.8 Register Definition ..... 625
33.8.1 WDOG_CS (Watchdog Control and Status Register) ..... 625
33.8.2 WDOG_COUNTER (Watchdog Counter Register) ..... 628
33.8.3 WDOG_TIMEOUT (Watchdog Timeout Value Register) ..... 629
33.8.4 WDOG_WINDOW (Watchdog Window Register) ..... 630
**Chapter 34 Error Injection Module (EIM) ..... 631**
34.1 Revision History ..... 631
34.2 Chip-specific Information ..... 632
34.3 Features ..... 632
34.4 Overview ..... 632
34.5 Functional Description ..... 633
34.6 Register Definition ..... 634
34.6.1 EIM_CR (Configuration Register) ..... 634
34.6.2 EIM_CTRL_REGn (Channel N Control Register) ..... 634
34.6.3 EIM_CPU0_LOCKSTEP (CPU0 LOCKSTEP Error Injection Register) ..... 636
34.6.4 EIM_DMA0_LOCKSTEP (DMA0 LOCKSTEP Error Injection Register) ..... 637
34.6.5 EIM_BUS_REGn (Bus Register) ..... 638
**Chapter 35 Error Reporting Module (ERM) ..... 640**
35.1 Revision History ..... 640
35.2 Chip-specific Information ..... 641
35.3 Features ..... 641
35.4 Overview ..... 641

Flagchip

35.5 Functional Description ..... 641
35.5.1 Initialization ..... 641
35.5.2 Single-bit Correction Events ..... 641
35.5.3 Non-correctable Error Events ..... 641
35.6 Register Definition ..... 642
35.6.1 ERM_CR0 (Configuration Register 0) ..... 642
35.6.2 ERM_CR1 (Configuration Register 1) ..... 644
35.6.3 ERM_CR2 (Configuration Register 2) ..... 646
35.6.4 ERM_CR3 (Configuration Register 3) ..... 648
35.6.5 ERM_SR0 (Status Register 0) ..... 650
35.6.6 ERM_SR1 (Status Register 1) ..... 653
35.6.7 ERM_SR2 (Status Register 2) ..... 655
35.6.8 ERM_SR3 (Status Register 3) ..... 658
35.6.9 ERM_EARn (Error Address Register) ..... 660
**Chapter 36 Interrupt Monitor (INTM) ..... 662**
36.1 Revision History ..... 662
36.2 Chip-specific Information ..... 663
36.3 Features ..... 663
36.4 Overview ..... 663
36.5 Functional Description ..... 663
36.5.1 INTM Register Access Mode ..... 663
36.5.2 INTM Interrupt Active Mode ..... 663
36.5.3 INTM Interrupt Inactive Mode ..... 663
36.6 Register Definition ..... 664
36.6.1 INTM_ER (Enable Register) ..... 664
36.6.2 INTM_IACKR (Interrupt Acknowledge Register) ..... 665
36.6.3 INTM_IRQSELRn (Interrupt Request Select Register) ..... 665
36.6.4 INTM_LATRn (Latency Register) ..... 666
36.6.5 INTM_TMRn (Timer Register) ..... 667
36.6.6 INTM_SRn (Status Register) ..... 667

Flagchip

**Chapter 37 Clock Monitor Unit (CMU) ..... 669**
37.1 Revision History ..... 669
37.2 Chip-specific Information ..... 670
37.3 Features ..... 670
37.4 Overview ..... 671
37.5 Block Diagram ..... 671
37.6 Functional Description ..... 671
37.6.1 Programming REF_WINDOW ..... 672
37.6.2 Programming MIN and MAX ..... 672
37.6.3 Special Mode of CMU ..... 673
37.6.4 CMU Programming Sequence ..... 673
37.7 Register Definition ..... 674
37.7.1 CMU_CTRL (Control Register) ..... 674
37.7.2 CMU_MIN (Minimum Threshold Register) ..... 675
37.7.3 CMU_MAX (Maximum Threshold Register) ..... 676
37.7.4 CMU_REF_WINDOW (Reference Window Register) ..... 676
37.7.5 CMU_MON_CNT (Monitor Counter Register) ..... 677
37.7.6 CMU_ST (Status Register) ..... 677
37.7.7 CMU_PERIOD (Period Monitor Mode Configuration Register) ..... 678
**Chapter 38 Interface Safety Monitor (ISM) ..... 679**
38.1 Revision History ..... 679
38.2 Chip-specific Information ..... 680
38.3 Features ..... 680
38.4 Overview ..... 680
38.5 Block Diagram ..... 681
38.6 Functional Description ..... 681
38.6.1 Filter & Prescaler Channel (FPC) ..... 681
38.6.2 Logic Analyzer Module (LAM) ..... 683
38.6.3 Event Combiner Module (ECM) ..... 687
38.7 Register Definition ..... 687

Flagchip

38.7.1 ISM_PARAM (Parameter Register) ..... 687
38.7.2 ISM_CTRL (Control Register) ..... 688
38.7.3 ISM_E_STATUS (Event Status Register) ..... 689
38.7.4 ISM_E_CTRL (Event Control Register) ..... 689
38.7.5 ISM_EC_CTRL (Event Control Mode Control Register) ..... 690
38.7.6 ISM_FPC_STATUSn (FPC Status Register) ..... 691
38.7.7 ISM_FPC_CTRLn (FPC Control Register) ..... 692
38.7.8 ISM_FPC_CONFIGn (FPC Configuration Register) ..... 692
38.7.9 ISM_FPC_TIMERn (FPC Timer Register) ..... 693
38.7.10 ISM_LAM_STATUSn (LAM Status Register) ..... 694
38.7.11 ISM_LAM_CTRLn (LAM Control Register) ..... 694
38.7.12 ISM_LAM_CONFIGn (LAM Configuration Register) ..... 695
38.7.13 ISM_LAM_COUNTERn (LAM Counter Register) ..... 697
**Chapter 39 Fault Control and Safety Management Unit (FCSMU) ..... 698**
39.1 Revision History ..... 698
39.2 Chip-specific Information ..... 699
39.3 Features ..... 699
39.4 Overview ..... 699
39.5 Block Diagram ..... 699
39.6 Functional Description ..... 699
39.6.1 FCSMU State Switch ..... 700
39.6.2 FCSMU Interrupt and Functional Reset ..... 703
39.6.3 FCSMU Configuration CRC Check ..... 704
39.6.4 FCSMU Configuration Timeout ..... 706
39.6.5 FCSMU Fault Management ..... 706
39.7 Register Definition ..... 709
39.7.1 FCSMU_CTRL (Control Register) ..... 710
39.7.2 FCSMU_OPRK (Operation Key Register) ..... 711
39.7.3 FCSMU_SOCTRL (Status Output Control Register) ..... 711
39.7.4 FCSMU_FCCR0 (Fault Clear Control Register 0) ..... 713

Flagchip

39.7.5 FCSMU_FRST0 (Fault Reset Control Register 0) ..... 713
39.7.6 FCSMU_FST0 (Fault Status Register) ..... 714
39.7.7 FCSMU_FST_UNLK (Fault Status Register Unlock Register) ..... 715
39.7.8 FCSMU_FE0 (Fault Enable Register) ..... 715
39.7.9 FCSMU_WARNING_EN0 (Warning State Enable Register 0) ..... 716
39.7.10 FCSMU_WARNING_TO (Warning Timeout Interval Register) ..... 716
39.7.11 FCSMU_CFG_TO (Configuration State Timeout Interval Register) ..... 717
39.7.12 FCSMU_SOUT_DIAG (SOUT Diagnostic Register) ..... 717
39.7.13 FCSMU_STATUS (Status Register) ..... 718
39.7.14 FCSMU_NTW (Normal to Warning Register) ..... 719
39.7.15 FCSMU_WTF (Warning to Fault Register) ..... 720
39.7.16 FCSMU_NTF (Normal to Fault Register) ..... 721
39.7.17 FCSMU_FTW (Fault to Warning Register) ..... 722
39.7.18 FCSMU_INJECT (Fault Injection Register) ..... 723
39.7.19 FCSMU_IRQ_STAT (IRQ Status Register) ..... 723
39.7.20 FCSMU_IRQ_EN (IRQ Enable Register) ..... 724
39.7.21 FCSMU_TEMP_UNLK (Temporary Configuration State Unlock Register) ..... 724
39.7.22 FCSMU_PERMNT_LOCK (Permanent Configuration State Lock Register) ..... 725
39.7.23 FCSMU_STMR (SOUT Timer Interval Register) ..... 726
39.7.24 FCSMU_WARNING_IEN0 (Warning State Interrupt Enable Register) ..... 726
39.7.25 FCSMU_FAULT_IEN0 (Fault State Interrupt Enable Register) ..... 727
39.7.26 FCSMU_SOUT_EN0 (SOUT Enable Register) ..... 728
39.7.27 FCSMU_WARNING_TMR (Warning State Timer Register) ..... 728
39.7.28 FCSMU_SM_TMR (Safe Mode State Timer Register) ..... 729
39.7.29 FCSMU_CFG_TMR (Configuration State Timer Register) ..... 729
39.7.30 FCSMU_SOUT_TMR (SOUT Timer Register) ..... 730
39.7.31 FCSMU_CRC_CTRL (CRC Control Register) ..... 730
39.7.32 FCSMU_CRC_RES (CRC Result Register) ..... 731
**Chapter 40 Safety Test and Control Unit (STCU) ..... 733**
40.1 Revision History ..... 733

Flagchip

40.2 Chip-specific Information ..... 734
40.3 Features ..... 734
40.4 Overview ..... 734
40.5 Functional Description ..... 734
40.5.1 Power-on Self-test ..... 734
40.5.2 Software Self-test ..... 734
40.5.3 Chip Status during Self-test ..... 735
40.5.4 Self-Test Sequence ..... 735
40.5.5 Pin Reset Control ..... 735
40.5.6 FCSMU Reset Control ..... 735
40.5.7 Watchdog Timer ..... 736
40.5.8 STCU Self-check ..... 736
40.5.9 NVR Data Load Check ..... 736
40.5.10 Clock Source ..... 736
40.5.11 LBIST ..... 736
40.5.12 MBIST ..... 736
40.5.13 Interruption ..... 737
40.5.14 Hardware SRAM Initialization ..... 737
40.6 Register Definition ..... 737
40.6.1 STCU_SELF_TEST_KEY (Self Test Key Register) ..... 738
40.6.2 STCU_SELF_TEST_CTRL (Self Test Control Register) ..... 738
40.6.3 STCU_SELF_TEST_STATUS (Self Test Status Register) ..... 740
40.6.4 STCU_SELF_TEST_TRIG_A (Self Test Trigger A Register) ..... 741
40.6.5 STCU_SELF_TEST_TRIG_B (Self Test Trigger B Register) ..... 742
40.6.6 STCU_SELF_TEST_RESET (Self Test Reset Register) ..... 743
40.6.7 STCU_LBIST_PAT_CTRL (LBIST Pattern Control Register) ..... 743
40.6.8 STCU_LBIST_EXP_MISR (LBIST Expected MISR Register) ..... 744
40.6.9 STCU_LBIST_ACT_MISR (LBIST Actual MISR Register) ..... 744
40.6.10 STCU_LBIST_STATUS (LBIST Status Register) ..... 745
40.6.11 STCU_MBIST_SEL (MBIST Select Register) ..... 745

Flagchip

40.6.12 STCU_MBIST_ALG (MBIST Algorithm Register) ..... 746
40.6.13 STCU_MBIST_DONE_STATUS (MBIST Done Status Register) ..... 747
40.6.14 STCU_MBIST_FAIL_STATUS (MBIST Fail Status Register) ..... 749
40.6.15 STCU_USER_GP (User General Purpose Register) ..... 750
40.6.16 STCU_SRAM_INI_CTRL (SRAM Initialization Control Register) ..... 750
40.6.17 STCU_SRAM_INI_STATUS (SRAM Initialization Status Register) ..... 751
40.6.18 STCU_SRAM_INI_SEL (SRAM Initialization Select Register) ..... 752
40.6.19 STCU_SRAM_INI_DONE_STATUS (SRAM Initialization Done Status Register) ..... 753
40.6.20 STCU_IRQ (Interrupt Request Register) ..... 754
**Chapter 41 Cyclic Redundancy Check (CRC) ..... 755**
41.1 Revision History ..... 755
41.2 Chip-specific Information ..... 756
41.3 Introduction ..... 756
41.4 Features ..... 756
41.5 Functional Description ..... 756
41.5.1 CRC Initialization/Reinitialization ..... 756
41.5.2 CRC Calculations ..... 756
41.5.3 8-bit CRC/16-bit CRC/32-bit CRC ..... 756
41.5.4 Data Swap Feature ..... 757
41.5.5 Types of Data Swap ..... 757
41.5.6 CRC Result Complement ..... 757
41.6 Register Definition ..... 757
41.6.1 CRC_DATA (Data Register) ..... 757
41.6.2 CRC_POLY (Polynomial Register) ..... 758
41.6.3 CRC_CR (Control Register) ..... 759
41.6.4 CRC_SDAT (Data Bit Swap Register) ..... 760
**Chapter 42 General-Purpose Input/Output (GPIO) ..... 761**
42.1 Revision History ..... 761
42.2 Features ..... 762
42.3 Overview ..... 762

Flagchip

42.4 Functional Description ..... 762
42.4.1 Domain Write Protection (DWP) ..... 762
42.4.2 General-Purpose Input ..... 762
42.4.3 General-Purpose Output ..... 762
42.5 Register Definition ..... 763
42.5.1 GPIO_PDOR (Port Data Output Register) ..... 763
42.5.2 GPIO_PSOR (Port Set Output Register) ..... 764
42.5.3 GPIO_PCOR (Port Clear Output Register) ..... 764
42.5.4 GPIO_PTOR (Port Toggle Output Register) ..... 765
42.5.5 GPIO_PDIR (Port Data Input Register) ..... 765
42.5.6 GPIO_PDDR (Port Data Direction Register) ..... 766
42.5.7 GPIO_PIDR (Port Input Disable Register) ..... 767
**Chapter 43 Port Control (PORT) ..... 768**
43.1 Revision History ..... 768
43.2 Chip-specific Information ..... 769
43.3 Features ..... 769
43.4 Overview ..... 769
43.5 Functional Description ..... 769
43.5.1 Domain Write Protection ..... 769
43.5.2 Pin Control ..... 770
43.5.3 Global Pin Control ..... 770
43.5.4 Global Interrupt Control ..... 770
43.5.5 External Interrupts ..... 770
43.5.6 Digital Filter ..... 770
43.5.7 Safety Related ..... 770
43.6 Register Definition ..... 771
43.6.1 PORT_PCRn (Port Control Register) ..... 771
43.6.2 PORT_GPCLR (Global Pin Control Low Register) ..... 774
43.6.3 PORT_GPCHR (Global Pin Control High Register) ..... 775
43.6.4 PORT_GICLR (Global Interrupt Control Low Register) ..... 776

Flagchip

43.6.5 PORT_GICHR (Global Interrupt Control High Register) ..... 776
43.6.6 PORT_ISFR (Interrupt Status Flag Register) ..... 777
43.6.7 PORT_DFER (Digital Filter Enable Register) ..... 777
43.6.8 PORT_DFCR (Digital Filter Clock Register) ..... 778
43.6.9 PORT_DFWR (Digital Filter Width Register) ..... 779
43.6.10 PORT_GLDWP (Global Domain Write Protection Register) ..... 779
**Chapter 44 Trigger Select (TRGSEL) ..... 781**
44.1 Revision History ..... 781
44.2 Chip-specific information ..... 782
44.3 Features ..... 782
44.4 Overview ..... 782
44.5 Block Diagram ..... 782
44.6 Register Definition ..... 783
44.6.1 TRGSEL_OUT_SELn (OUT SEL for N) ..... 783
**Chapter 45 Logic Unit (LU) ..... 785**
45.1 Revision History ..... 785
45.2 Chip-specific Information ..... 786
45.3 Features ..... 786
45.4 Overview ..... 786
45.5 Block Diagram ..... 786
45.6 Functional Description ..... 787
45.6.1 Input Mode Configuration ..... 787
45.6.2 Output Mode Configuration ..... 787
45.6.3 LU Timing Between Inputs and Outputs ..... 788
45.7 Register Definition ..... 789
45.7.1 LU_AOI_0n (AOI0 Configuration Register) ..... 789
45.7.2 LU_AOI_1n (AOI1 Configuration Register) ..... 791
45.7.3 LU_CTRLn (LG Configuration Register) ..... 794
45.7.4 LU_FILTn (Input filter Register) ..... 795
**Chapter 46 Analog-to-Digital Converter (ADC) ..... 797**

Flagchip

46.1 Revision History ..... 797
46.2 Chip-specific Information ..... 798
46.3 Features ..... 798
46.4 Overview ..... 799
46.5 Block Diagram ..... 799
46.6 Functional Description ..... 799
46.6.1 ADC Control ..... 799
46.6.2 Trigger Detection ..... 803
46.6.3 Conversion Sequence Mode ..... 805
46.6.4 Data Management ..... 808
46.6.5 ADC Interrupts ..... 810
46.6.6 Low Power Features ..... 810
46.6.7 ADC Compare ..... 811
46.6.8 ADC Clock ..... 811
46.7 Register Definition ..... 811
46.7.1 ADC_INT_STATUS (Interrupt Status Register) ..... 811
46.7.2 ADC_INT_ENABLE (Interrupt Enable Register) ..... 814
46.7.3 ADC_CONTROL (Control Register) ..... 816
46.7.4 ADC_CFG1 (Configuration1 Register) ..... 817
46.7.5 ADC_CFG2 (Configuration2 Register) ..... 819
46.7.6 ADC_SMPR (Sampling Rate Register) ..... 821
46.7.7 ADC_CMP_CTRL (Compare Control Register) ..... 822
46.7.8 ADC_CMP_TR (Compare Threshold Register) ..... 823
46.7.9 ADC_SGCSRn (Sequence Group Control and Status Register) ..... 824
46.7.10 ADC_CAL (Calibration Register) ..... 825
46.7.11 ADC_FIFO_DATA (FIFO Data Register) ..... 825
46.7.12 ADC_SCn (Sequence Configuration Register) ..... 826
46.7.13 ADC_RESULTn (Result Register) ..... 827
**Chapter 47 Comparator (CMP) ..... 829**
47.1 Revision History ..... 829

Flagchip

47.2 Chip-specific Information ..... 830
47.3 Features ..... 830
47.4 Overview ..... 831
47.4.1 Instantiation Information ..... 831
47.4.2 CMP Analog Block Diagram ..... 831
47.4.3 DAC Reference ..... 832
47.4.4 Window Control ..... 832
47.5 Functional Description ..... 832
47.5.1 Function Mode ..... 832
47.5.2 Initialization ..... 834
47.5.3 Enabling Filter Modes ..... 834
47.5.4 Interrupts ..... 834
47.5.5 DMA Support ..... 834
47.5.6 Channel Scan Mode ..... 835
47.5.7 Legal Channel Decode ..... 836
47.6 Register Definition ..... 837
47.6.1 CMP_CCR0 (Comparator Control Register 0) ..... 837
47.6.2 CMP_CCR1 (Comparator Control Register 1) ..... 838
47.6.3 CMP_CCR2 (Comparator Control Register 2) ..... 840
47.6.4 CMP_CCR3 (Comparator Control Register 3) ..... 842
47.6.5 CMP_DCR (Comparator DAC Control Register) ..... 842
47.6.6 CMP_IER (Comparator Interrupt Enable Register) ..... 843
47.6.7 CMP_CSR (Comparator Status Register) ..... 844
47.6.8 CMP_CSCR0 (Channel Scan Control Register 0) ..... 845
47.6.9 CMP_CSCR1 (Channel Scan Control Register 1) ..... 846
47.6.10 CMP_CSCSR (Channel Scan Control and Status Register) ..... 848
47.6.11 CMP_CSSR (Channel Scan Status Register) ..... 850
**Chapter 48 Temperature Monitor Unit (TMU) ..... 852**
48.1 Revision History ..... 852
48.2 Chip-specific Information ..... 853

Flagchip

48.3 Features ..... 853
48.4 Overview ..... 853
48.4.1 Voltage-based Temperature Sensor ..... 853
48.4.2 Flag-based Temperature Sensor ..... 853
48.5 Block Diagram ..... 853
48.6 Functional Description ..... 854
48.6.1 Lock and Unlock ..... 854
48.6.2 Flag-based Temperature Sensor ..... 854
48.6.3 Voltage-based Temperature Sensor ..... 855
48.7 Register Definition ..... 856
48.7.1 TMU_UNLOCK (Unlock Register) ..... 856
48.7.2 TMU_TF_CTRL (Temperature Flag Control Register) ..... 857
48.7.3 TMU_TF_STATUS (Temperature Flag Status Register) ..... 858
48.7.4 TMU_TV_CTRL (Temperature Voltage Control Register) ..... 859
48.7.5 TMU_TV_STATUS (Temperature Voltage Status Register) ..... 860
48.7.6 TMU_TV_TRIM (Temperature Voltage Trim Register) ..... 861
**Chapter 49 Flexible Timer Unit (FTU) ..... 862**
49.1 Revision History ..... 862
49.2 Chip-specific Information ..... 863
49.2.1 FTU Fault Detection Inputs ..... 863
49.2.2 FTU Hardware Triggers and Synchronization ..... 863
49.2.3 FTU Input Capture Options ..... 863
49.2.4 FTU Output Modulation ..... 863
49.2.5 FTU Global Time Base (GTB) ..... 863
49.2.6 FTU Counter Width ..... 864
49.3 Features ..... 864
49.4 Block Diagram ..... 865
49.5 Functional Description ..... 865
49.5.1 Counter Mode and Events ..... 865
49.5.2 Channel Operation Mode ..... 867

Flagchip

49.5.3 Quadrature Decoder Mode ..... 875
49.5.4 Registers Updated from Write Buffers ..... 876
49.5.5 Synchronization ..... 878
49.5.6 Deadtime Insertion ..... 879
49.5.7 Software Output Control Mode ..... 880
49.5.8 Fault Control ..... 881
49.5.9 Trigger Output Support ..... 883
49.5.10 Debug Mode ..... 884
49.5.11 Reload Points ..... 885
49.5.12 Channel Signal Related ..... 886
49.5.13 Interrupts ..... 887
49.5.14 DMA ..... 888
49.6 Register Definition ..... 889
49.6.1 FTU_SC (Status and Control Register) ..... 890
49.6.2 FTU_CNT (Counter Register) ..... 892
49.6.3 FTU_MOD (Modulo Register) ..... 893
49.6.4 FTU_CnSC (Channel n Status and Control Register) ..... 893
49.6.5 FTU_CnV (Channel n Value Register) ..... 896
49.6.6 FTU_CNTIN (Counter Initial Value Register) ..... 896
49.6.7 FTU_STATUS (Capture and Compare Status Register) ..... 897
49.6.8 FTU_MODE (Mode Selection Register) ..... 898
49.6.9 FTU_SYNC (Synchronization Register) ..... 899
49.6.10 FTU_OUTINIT (Initial State for Channels Output Register) ..... 901
49.6.11 FTU_OUTMASK (Output Mask Register) ..... 902
49.6.12 FTU_CHCTRL (Channel Control Register) ..... 902
49.6.13 FTU_DEADTIME (Deadtime Configuration Register) ..... 906
49.6.14 FTU_TRIGCONF (FTU Trigger Out Configuration Register) ..... 907
49.6.15 FTU_POL (Polarity Register) ..... 908
49.6.16 FTU_FMS (Fault Mode Status Register) ..... 909
49.6.17 FTU_FILTER (Input Capture Filter Control Register) ..... 910

Flagchip

49.6.18 FTU_FLTCTRL (Fault Control Register) ..... 911
49.6.19 FTU_QDCTRL (Quadrature Decoder Control and Status Register) ..... 912
49.6.20 FTU_CONF (Configuration Register) ..... 913
49.6.21 FTU_FLTPOL (FTU Fault Input Polarity Register) ..... 914
49.6.22 FTU_SYNCONF (Synchronization Configuration Register) ..... 916
49.6.23 FTU_INVCTRL (FTU Inverting Control Register) ..... 916
49.6.24 FTU_SWOCTRL (FTU Software Output Control Register) ..... 917
49.6.25 FTU_PWMLOAD (FTU PWM Load Register) ..... 918
49.6.26 FTU_PAIRnDEADTIME (Pair (n) Deadtime Configuration Register) ..... 918
**Chapter 50 FC Programmable Interrupt Timer (FCPIT) ..... 920**
50.1 Revision History ..... 920
50.2 Chip-specific Information ..... 921
50.2.1 FCPIT/DMA Trigger ..... 921
50.3 Overview ..... 921
50.3.1 Modes of Operation ..... 921
50.4 Block Diagram ..... 922
50.5 Features ..... 922
50.6 Functional Description ..... 922
50.6.1 Initialization ..... 922
50.6.2 Timer Modes ..... 923
50.6.3 Timers Control from Trigger ..... 923
50.6.4 Channel-Chained Function ..... 924
50.7 Register Definition ..... 924
50.7.1 FCPIT_MCR (Module Control Register) ..... 924
50.7.2 FCPIT_MSR (Module Status Register) ..... 925
50.7.3 FCPIT_MIER (Module Interrupt Enable Register) ..... 926
50.7.4 FCPIT_SETTEN (Set Timer Enable Register) ..... 927
50.7.5 FCPIT_CLRTEN (Clear Timer Enable Register) ..... 929
50.7.6 FCPIT_TVALn (Timer Value Register) ..... 930
50.7.7 FCPIT_CVALn (Current Timer Value Register) ..... 930

Flagchip

50.7.8 FCPIT_TCTRLn (Timer Control Register) ..... 931
**Chapter 51 Timer Stamp (TSTMP) ..... 934**
51.1 Revision History ..... 934
51.2 Chip-specific Information ..... 935
51.3 Features ..... 935
51.4 Overview ..... 935
51.5 Functional Description ..... 935
51.5.1 Modulate Timer Clock Source ..... 935
51.5.2 TSTMP Trigger ..... 935
51.6 Register Definition ..... 936
51.6.1 TSTMP_VALL (Low Value Register) ..... 936
51.6.2 TSTMP_VALH (High Value Register) ..... 936
51.6.3 TSTMP_MOD_INTEN (Modulate Interrupt Enable Register) ..... 937
51.6.4 TSTMP_MOD_STATUS (Modulate Status Register) ..... 939
51.6.5 TSTMP_MOD0_SETVAL (Modulate0 Set Value Register) ..... 940
51.6.6 TSTMP_MOD1_SETVAL (Modulate1 Set Value Register) ..... 940
51.6.7 TSTMP_MOD2_SETVAL (Modulate2 Set Value Register) ..... 941
51.6.8 TSTMP_MOD3_SETVAL (Modulate3 Set Value Register) ..... 941
**Chapter 52 Real-Time Clock (RTC) ..... 943**
52.1 Revision History ..... 943
52.2 Chip-specific Information ..... 944
52.3 Features ..... 944
52.4 Overview ..... 944
52.5 Functional Description ..... 944
52.5.1 Clock ..... 944
52.5.2 Time Counter ..... 944
52.5.3 Interrupt ..... 945
52.5.4 Register Lock ..... 945
52.6 Register Definition ..... 945
52.6.1 RTC_SR (Seconds Register) ..... 945

Flagchip

52.6.2 RTC_PR (Prescaler Register) ..... 946
52.6.3 RTC_AR (Alarm Register) ..... 947
52.6.4 RTC_COMPR (Compensation Register) ..... 947
52.6.5 RTC_CR (Control Register) ..... 948
52.6.6 RTC_STR (Status Register) ..... 949
52.6.7 RTC_LR (Lock Register) ..... 950
52.6.8 RTC_IER (Interrupt Enable Register) ..... 951
**Chapter 53 Always-on Timer (AONTIMER) ..... 953**
53.1 Revision History ..... 953
53.2 Chip-specific Information ..... 954
53.3 Features ..... 954
53.4 Overview ..... 954
53.5 Functional Description ..... 954
53.5.1 AONTIMER Clocking ..... 954
53.5.2 AONTIMER Prescaler ..... 954
53.5.3 AONTIMER Glitch Filter ..... 954
53.5.4 AONTIMER Counter ..... 955
53.5.5 AONTIMER Compare ..... 955
53.5.6 AONTIMER Interrupt ..... 955
53.5.7 AONTIMER DMA Request ..... 955
53.5.8 AONTIMER Hardware Trigger ..... 955
53.6 Register Definition ..... 956
53.6.1 AONTIMER_CSR (Control Status Register) ..... 956
53.6.2 AONTIMER_PSR (Prescale Register) ..... 957
53.6.3 AONTIMER_CMR (Compare Register) ..... 958
53.6.4 AONTIMER_CNR (Counter Register) ..... 959
**Chapter 54 Programmable Timer (PTIMER) ..... 960**
54.1 Revision History ..... 960
54.2 Chip-specific Information ..... 961
54.3 Features ..... 961

Flagchip

54.4 Overview ..... 961
54.4.1 Instantiation Information ..... 961
54.4.2 PTIMER Trigger Output Connections with ADC and TRGSEL ..... 961
54.4.3 PTIMER Software Trigger Source ..... 961
54.4.4 PTIMER Trigger Input from TRGSEL ..... 961
54.4.5 Back-to-back Operation Mode with ADC ..... 961
54.5 Functional Description ..... 962
54.5.1 PTIMER Pre-trigger and Trigger Outputs ..... 962
54.5.2 PTIMER Triggering ADC ..... 962
54.5.3 PTIMER Functions ..... 965
54.6 Register Definition ..... 966
54.6.1 PTIMER_STATUS_CTRL (Status and Control Register) ..... 966
54.6.2 PTIMER_MAX_CNT (Max Count Number Register) ..... 969
54.6.3 PTIMER_CNT (Counter Register) ..... 970
54.6.4 PTIMER_INT_DLY (Interrupt Delay Register) ..... 971
54.6.5 PTIMER_CTRL_CHn (Channel n Control Register) ..... 971
54.6.6 PTIMER_STATUS_CHn (Channel n Status Register) ..... 972
54.6.7 PTIMER_DLY0_CHn (Channel n Delay 0 Register) ..... 973
54.6.8 PTIMER_DLY1_CHn (Channel n Delay 1 Register) ..... 974
54.6.9 PTIMER_DLY2_CHn (Channel n Delay 2 Register) ..... 974
54.6.10 PTIMER_DLY3_CHn (Channel n Delay 3 Register) ..... 975
54.6.11 PTIMER_DLY4_CHn (Channel n Delay 4 Register) ..... 976
54.6.12 PTIMER_DLY5_CHn (Channel n Delay 5 Register) ..... 976
54.6.13 PTIMER_DLY6_CHn (Channel n Delay 6 Register) ..... 977
54.6.14 PTIMER_DLY7_CHn (Channel n Delay 7 Register) ..... 978
54.6.15 PTIMER_POEN (Pulse-Out Enable Register) ..... 978
54.6.16 PTIMER_PODLY (Pulse-Out Delay Register) ..... 979
**Chapter 55 Frequency Measurement (FREQM) ..... 981**
55.1 Revision History ..... 981
55.2 Chip-specific Information ..... 982

Flagchip

55.3 Features ..... 982
55.4 Overview ..... 982
55.5 Block Diagram ..... 982
55.6 Functional Description ..... 982
55.6.1 Measured Clock Selection and Divider ..... 982
55.6.2 Control Sequence ..... 982
55.6.3 Interrupt ..... 983
55.6.4 Reset ..... 983
55.6.5 Stop Mode ..... 983
55.7 Register Definition ..... 984
55.7.1 FREQM_CTRL (Control Register) ..... 984
55.7.2 FREQM_MES_CNT (Measure Counter Register) ..... 986
55.7.3 FREQM_REF_CNT (Reference Counter Register) ..... 986
55.7.4 FREQM_MES_LENGTH (Measure Counter Length Register) ..... 987
55.7.5 FREQM_REF_TIMEOUT (Reference Counter Timeout Register) ..... 987
55.7.6 FREQM_CNT_STATUS (Counter Status Register) ..... 988
55.7.7 FREQM_REF_CNT_SAVE (Saved Reference Counter Register) ..... 988
**Chapter 56 Time Processing Unit (TPU) ..... 990**
56.1 Revision History ..... 990
56.2 Chip-specific Information ..... 991
56.2.1 Chip Mode and TPU Register Side ..... 991
56.2.2 TPU Clocking ..... 991
56.2.3 TPU Reset ..... 991
56.2.4 TPU Debug ..... 991
56.2.5 TPU Global Time Base (GTB) ..... 991
56.2.6 TPU Channel Input ..... 991
56.2.7 TPU Channel Output ..... 992
56.3 Features ..... 992
56.4 Overview ..... 992
56.5 Functional Description ..... 995

Flagchip

56.5.1 System Mode and Resource ..... 995
56.5.2 Function and ISR ..... 996
56.5.3 Enhanced Channels ..... 1000
56.5.4 Time Base ..... 1019
56.5.5 EAC-TPU Angle Counter ..... 1023
56.6 TPU_H Register Definition ..... 1035
56.6.1 TPU_H_GCR_VSR (Global Control Region -Visible Status Register) ..... 1036
56.6.2 TPU_H_GCR_VIR (Global Control Region - Visible Input Status Register) ..... 1037
56.6.3 TPU_H_GCR_VOR (Global Control Region - Visible Output Status Register) ..... 1037
56.6.4 TPU_H_GCR_VOBR (Global Control Region - Visible Output Enable Status Register) ..... 1038
56.6.5 TPU_H_GCR_VM1R (Global Control Region - Visible MRL1 Register) ..... 1038
56.6.6 TPU_H_GCR_VM2R (Global Control Region -Visible MRL2 Register) ..... 1039
56.6.7 TPU_H_GCR_VT1R (Global Control Region - Visible TDL1 Register) ..... 1039
56.6.8 TPU_H_GCR_VT2R (Global Control Region - Visible TDL2 Register) ..... 1040
56.6.9 TPU_H_GCR_EM1R (Global Control Region - Visible Event by MRL1 Register) ..... 1040
56.6.10 TPU_H_GCR_EM2R (Global Control Region - Visible Event by MRL2 Register) ..... 1041
56.6.11 TPU_H_GCR_ET1R (Global Control Region - Visible Event by TDL1 Register) ..... 1041
56.6.12 TPU_H_GCR_ET2R (Global Control Region - Visible Event by TDL2 Register) ..... 1042
56.6.13 TPU_H_GCR_VHSR (Global Control Region - Visible Host Acknowledge Register) ..... 1042
56.6.14 TPU_H_TBR_CR (Time Bases Region - Control Register) ..... 1043
56.6.15 TPU_H_TBR_T1R (Time Base Region - TCR1 Value Register) ..... 1045
56.6.16 TPU_H_TBR_T2R (Time Base Region - TCR2 Value Register) ..... 1046
56.6.17 TPU_H_TBR_TPR (Time Base Region - Tooth Program Register) ..... 1046
56.6.18 TPU_H_TBR_TRR (Time Base Region - Tick Rate Register) ..... 1047
56.6.19 TPU_H_TBR_T1MR (Time Base Region - TCR1 Maximum Register) ..... 1048
56.6.20 TPU_H_TBR_T2MR (Time Base Region - TCR2 Maximum Register) ..... 1048
56.6.21 TPU_H_CHn_CR (Channel N Control Register) ..... 1049
56.6.22 TPU_H_CHn_SR (Channel N Status Register) ..... 1050
56.6.23 TPU_H_CHn_SCR (Channel N Status Control Register) ..... 1051
56.6.24 TPU_H_CHn_EFR (Channel N Event and Flag Register) ..... 1052

Flagchip

56.7 TPU_E Register Definition ..... 1054
56.7.1 TPU_E_GCR_CR (Global Control Region - Control Register) ..... 1055
56.7.2 TPU_E_GCR_SRR (Global Control Region - Software Reset Register) ..... 1057
56.7.3 TPU_E_GCR_VSR (Global Control Region - Visible Status Register) ..... 1057
56.7.4 TPU_E_GCR_VIR (Global Control Region - Visible Input Status Register) ..... 1058
56.7.5 TPU_E_GCR_VOR (Global Control Region -Visible Output Status Register) ..... 1058
56.7.6 TPU_E_GCR_VOBR (Global Control Region - Visible Output Enable Status Register) ..... 1059
56.7.7 TPU_E_GCR_VM1R (Global Control Region - Visible MRL1 Register) ..... 1059
56.7.8 TPU_E_GCR_VM2R (Global Control Region - Visible MRL2 Register) ..... 1060
56.7.9 TPU_E_GCR_VT1R (Global Control Region - Visible TDL1 Register) ..... 1060
56.7.10 TPU_E_GCR_VT2R (Global Control Region - Visible TDL2 Register) ..... 1061
56.7.11 TPU_E_GCR_EM1R (Global Control Region - Visible Event by MRL1 Register) ..... 1061
56.7.12 TPU_E_GCR_EM2R (Global Control Region - Visible Event by MRL2 Register) ..... 1062
56.7.13 TPU_E_GCR_ET1R (Global Control Region - Visible Event by TDL1 Register) ..... 1062
56.7.14 TPU_E_GCR_ET2R (Global Control Region - Visible Event by TDL2 Register) ..... 1063
56.7.15 TPU_E_GCR_VHSR (Global Control Region - Visible Host Request Register) ..... 1063
56.7.16 TPU_E_GCR_AFGR (Global Control Region - Interrupt Arbiter Fliter Grade Register) ..... 1064
56.7.17 TPU_E_GCR_ASR (Global Control Region - Interrupt Arbiter Status Register) ..... 1064
56.7.18 TPU_E_GCR_AID (Global Control Region - Interrupt Arbiter ID Register) ..... 1065
56.7.19 TPU_E_TBR_CR (Time Bases Region - Control Register) ..... 1066
56.7.20 TPU_E_TBR_T1R (Time Base Region - TCR1 Value Register) ..... 1068
56.7.21 TPU_E_TBR_T2R (Time Base Region - TCR2 Value Register) ..... 1069
56.7.22 TPU_E_TBR_TPR (Time Base Region - Tooth Program Register) ..... 1069
56.7.23 TPU_E_TBR_TRR (Time Base Region - Tick Rate Register) ..... 1070
56.7.24 TPU_E_TBR_T1MR (Time Base Region - TCR1 Maximum Register) ..... 1071
56.7.25 TPU_E_TBR_T2MR (Time Base Region - TCR2 Maximum Register) ..... 1071
56.7.26 TPU_E_CHn_CR (Channel N Control Register) ..... 1072
56.7.27 TPU_E_CHn_SR (Channel N Status Register) ..... 1073
56.7.28 TPU_E_CHn_SCR (Channel N Status Control Register) ..... 1074
56.7.29 TPU_E_CHn_EFR (Channel N Event and Flag Register) ..... 1075

Flagchip

56.7.30 TPU_E_CHn_CR2 (Channel N Control Register 2) ..... 1077
56.7.31 TPU_E_CHn_MR (Channel N Mode configuration visible Register) ..... 1079
56.7.32 TPU_E_CHn_ER1 (Channel N Event Control Register or Part 1) ..... 1080
56.7.33 TPU_E_CHn_ER2 (Channel N Event Control Register or Part 2) ..... 1080
56.7.34 TPU_E_CHn_ECR (Channel N Event Clear Register) ..... 1081
56.7.35 TPU_E_CHn_OCR (Channel N Output User Control Register) ..... 1082
**Chapter 57 FC Serial Peripheral Interface (FCSPI) ..... 1084**
57.1 Revision History ..... 1084
57.2 Chip-specific Information ..... 1085
57.3 Features ..... 1085
57.4 Overview ..... 1085
57.5 Functional Description ..... 1086
57.5.1 Software Reset ..... 1086
57.5.2 New Transfer in Master Mode ..... 1086
57.5.3 New Transfer in Slave Mode ..... 1087
57.5.4 Parallel Transfer (in Master Mode) ..... 1087
57.5.5 Transfer Monitor ..... 1087
57.5.6 PCS-SCK Configuration ..... 1088
57.6 Register Definition ..... 1088
57.6.1 FCSPI_CTRL (Control Register) ..... 1089
57.6.2 FCSPI_STATUS (Status Register) ..... 1090
57.6.3 FCSPI_INT_EN (Interrupt Enable Register) ..... 1091
57.6.4 FCSPI_DMA_EN (DMA Enable Register) ..... 1092
57.6.5 FCSPI_CFG0 (Configuration Register 0) ..... 1093
57.6.6 FCSPI_CFG1 (Configuration Register 1) ..... 1094
57.6.7 FCSPI_DATA_MATCH0 (Data Match Register 0) ..... 1095
57.6.8 FCSPI_DATA_MATCH1 (Data Match Register 1) ..... 1096
57.6.9 FCSPI_CLK_CFG (Clock Configuration Register) ..... 1096
57.6.10 FCSPI_FIFO_WTM (FIFO Water Mark Register) ..... 1097
57.6.11 FCSPI_FIFO_STATUS (FIFO Status Register) ..... 1098

Flagchip

57.6.12 FCSPI_TR_CTRL (Transmit and Receive Control Register) ..... 1098
57.6.13 FCSPI_TX_DATA (Transmit Data Register) ..... 1101
57.6.14 FCSPI_RX_STATUS (Receive Status Register) ..... 1102
57.6.15 FCSPI_RX_DATA (Receive Data Register) ..... 1102
**Chapter 58 FC Inter-Integrated Circuit (FCIIC) ..... 1104**
58.1 Revision History ..... 1104
58.2 Chip-specific Information ..... 1105
58.3 Overview ..... 1105
58.4 Block Diagram ..... 1105
58.4.1 Modes of Operation ..... 1106
58.5 Features ..... 1106
58.6 Functional Description ..... 1106
58.6.1 Bus Speed ..... 1106
58.6.2 General Call, 7-bit and 10-bit Addressing ..... 1107
58.6.3 Master Mode ..... 1107
58.6.4 Slave Mode ..... 1109
58.6.5 Clock and Reset ..... 1110
58.6.6 Timing Parameters ..... 1111
58.6.7 Interrupts and DMA Requests ..... 1112
58.6.8 Peripheral Triggers ..... 1113
58.7 Register Definition ..... 1114
58.7.1 FCIIC_MCR (Master Control Register) ..... 1115
58.7.2 FCIIC_MSR (Master Status Register) ..... 1116
58.7.3 FCIIC_MIER (Master Interrupt Enable Register) ..... 1118
58.7.4 FCIIC_MDER (Master DMA Enable Register) ..... 1119
58.7.5 FCIIC_MCFGR0 (Master Configuration Register 0) ..... 1120
58.7.6 FCIIC_MCFGR1 (Master Configuration Register 1) ..... 1121
58.7.7 FCIIC_MCFGR2 (Master Configuration Register 2) ..... 1123
58.7.8 FCIIC_MCFGR3 (Master Configuration Register 3) ..... 1124
58.7.9 FCIIC_MDMR (Master Data Match Register) ..... 1124

Flagchip

58.7.10 FCIIC_MCCR (Master Clock Configuration Register) ..... 1125
58.7.11 FCIIC_MFCR (Master FIFO Control Register) ..... 1126
58.7.12 FCIIC_MFSR (Master FIFO Status Register) ..... 1127
58.7.13 FCIIC_MTDR (Master Transmit Data Register) ..... 1127
58.7.14 FCIIC_MRDR (Master Receive Data Register) ..... 1128
58.7.15 FCIIC_SCR (Slave Control Register) ..... 1129
58.7.16 FCIIC_SSR (Slave Status Register) ..... 1130
58.7.17 FCIIC_SIER (Slave Interrupt Enable Register) ..... 1132
58.7.18 FCIIC_SDER (Slave DMA Enable Register) ..... 1134
58.7.19 FCIIC_SCFGR1 (Slave Configuration Register 1) ..... 1135
58.7.20 FCIIC_SCFGR2 (Slave Configuration Register 2) ..... 1137
58.7.21 FCIIC_SAMR (Slave Address Match Register) ..... 1139
58.7.22 FCIIC_SASR (Slave Address Status Register) ..... 1139
58.7.23 FCIIC_STAR (Slave Transmit ACK Register) ..... 1140
58.7.24 FCIIC_STDR (Slave Transmit Data Register) ..... 1141
58.7.25 FCIIC_SRDR (Slave Receive Data Register) ..... 1142
**Chapter 59 FC Universal Asynchronous Receiver Transmitter (FCUART) ..... 1143**
59.1 Revision History ..... 1143
59.2 Features ..... 1144
59.3 Overview ..... 1144
59.3.1 Clocking and Resets ..... 1144
59.3.2 Baud Rate Generation ..... 1145
59.3.3 Receiver Function Description ..... 1145
59.3.4 Transmitter Function Description ..... 1147
59.3.5 Interrupts and Status Flags ..... 1149
59.3.6 Other FCUART Functions ..... 1149
59.3.7 Peripheral Triggers ..... 1150
59.4 Register Definition ..... 1151
59.4.1 FCUART_RST (Software Reset Register) ..... 1151
59.4.2 FCUART_PIN_CFG (Pin Configuration Register) ..... 1152

Flagchip

59.4.3 FCUART_BAUD (Baud Rate Register) ..... 1152
59.4.4 FCUART_STAT (Status Register) ..... 1155
59.4.5 FCUART_CTRL (Control Register) ..... 1159
59.4.6 FCUART_DATA (Data Register) ..... 1163
59.4.7 FCUART_MATCH (Match Address Register) ..... 1164
59.4.8 FCUART_MODIR (Modem IrDA Register) ..... 1165
59.4.9 FCUART_FIFO (FIFO Register) ..... 1167
59.4.10 FCUART_WATERMARK (Watermark Register) ..... 1169
**Chapter 60 FLEX Controller Area Network (FLEXCAN) ..... 1171**
60.1 Revision History ..... 1171
60.2 Chip-specific Information ..... 1172
60.2.1 CAN Embedded RAM Initialization ..... 1172
60.3 Features ..... 1172
60.4 Overview ..... 1173
60.4.1 Modes of Operation ..... 1173
60.4.2 Signal Descriptions ..... 1174
60.5 Block Diagram ..... 1175
60.6 Functional Description ..... 1175
60.6.1 Initialization Sequence ..... 1176
60.6.2 Transmit Process ..... 1177
60.6.3 Arbitration Process ..... 1178
60.6.4 Receive Process ..... 1180
60.6.5 Mask of ID Filter ..... 1182
60.6.6 Matching Process ..... 1183
60.6.7 Receive Process under Pretended Networking Mode ..... 1185
60.6.8 Move Process ..... 1188
60.6.9 Data Coherence ..... 1189
60.6.10 Enhanced Rx FIFO ..... 1191
60.6.11 Legacy RX FIFO ..... 1195
60.6.12 CAN Protocol Related Features ..... 1197

Flagchip

60.6.13 Modes of Operation Details ..... 1200
60.6.14 Interrupts ..... 1202
60.6.15 Bus interface ..... 1203
60.6.16 Detection and Correction of Memory Errors ..... 1204
60.6.17 Clock Domains and Restrictions ..... 1207
60.7 Register and RAM Special ..... 1207
60.7.1 Registers in RAM ..... 1207
60.7.2 Register Access and Reset ..... 1207
60.7.3 Registers Supported in Chip ..... 1209
60.7.4 Message Buffer Definition ..... 1211
60.7.5 Message Buffer Memory Map ..... 1216
60.7.6 Legacy Rx FIFO Structure ..... 1217
60.7.7 Enhanced Rx (ERX) FIFO Structure ..... 1218
60.7.8 Enhanced Rx FIFO ID Match Example ..... 1219
60.8 Register Definition ..... 1221
60.8.1 FLEXCAN_MCR (Module Configuration Register) ..... 1223
60.8.2 FLEXCAN_CTRL1 (Control 1 Register) ..... 1228
60.8.3 FLEXCAN_TIMER (Free Running Timer) ..... 1233
60.8.4 FLEXCAN_RXMGMASK (Rx Mailboxes Global Mask Register) ..... 1234
60.8.5 FLEXCAN_RX14MASK (Rx 14 Mask Register) ..... 1234
60.8.6 FLEXCAN_RX15MASK (Rx 15 Mask Register) ..... 1235
60.8.7 FLEXCAN_ECR (Error Counter Register) ..... 1235
60.8.8 FLEXCAN_ESR1 (Error and Status 1 Register) ..... 1236
60.8.9 FLEXCAN_IMASK1 (Interrupt Masks 1 Register) ..... 1241
60.8.10 FLEXCAN_IFLAG1 (Interrupt Flags 1 Register) ..... 1242
60.8.11 FLEXCAN_CTRL2 (Control 2 Register) ..... 1244
60.8.12 FLEXCAN_ESR2 (Error and Status 2 Register) ..... 1249
60.8.13 FLEXCAN_CRCR (CRC Register) ..... 1250
60.8.14 FLEXCAN_RXFGMASK (Legacy Rx FIFO Global Mask Register) ..... 1251
60.8.15 FLEXCAN_RXFIR (Legacy Rx FIFO Information Register) ..... 1251

Flagchip

60.8.16 FLEXCAN_CBT (CAN Bit Timing Register) ..... 1252
60.8.17 FLEXCAN_RAMn (Message Buffer Registers) ..... 1254
60.8.18 FLEXCAN_RXIMRn (Rx Individual Mask registers) ..... 1254
60.8.19 FLEXCAN_MECR (Memory Error Control Register) ..... 1255
60.8.20 FLEXCAN_ERRIAR (Error Injection Address Register) ..... 1257
60.8.21 FLEXCAN_ERRIDPR (Error Injection Data Pattern Register) ..... 1258
60.8.22 FLEXCAN_ERRIPPR (Error Injection Parity Pattern Register) ..... 1258
60.8.23 FLEXCAN_RERRAR (Error Report Address Register) ..... 1259
60.8.24 FLEXCAN_RERRDR (Error Report Data Register) ..... 1260
60.8.25 FLEXCAN_RERRSYNR (Error Report Syndrome Register) ..... 1260
60.8.26 FLEXCAN_ERRSR (Error Status Register) ..... 1261
60.8.27 FLEXCAN_CTRL1_PN (Pretended Networking Control 1 Register) ..... 1262
60.8.28 FLEXCAN_CTRL2_PN (Pretended Networking Control 2 Register) ..... 1264
60.8.29 FLEXCAN_WU_MTC (Pretended Networking Wake Up Match Register) ..... 1265
60.8.30 FLEXCAN_FLT_ID1 (Pretended Networking ID Filter 1 Register) ..... 1266
60.8.31 FLEXCAN_FLT_DLC (Pretended Networking DLC Filter Register) ..... 1267
60.8.32 FLEXCAN_PL1_LO (Pretended Networking Payload Low Filter 1 Register) ..... 1267
60.8.33 FLEXCAN_PL1_HI (Pretended Networking Payload High Filter 1 Register) ..... 1268
60.8.34 FLEXCAN_FLT_ID2_IDMASK (Pretended Networking ID Filter 2 Register/ ID Mask Register) ..... 1269
60.8.35 FLEXCAN_PL2_PLMASK_LO (Pretended Networking Payload Low Filter 2 Register/ Payload Low Mask Register) ..... 1270
60.8.36 FLEXCAN_PL2_PLMASK_HI (Pretended Networking Payload High Filter 2 Low Order Bits/ Payload High Mask Register) ..... 1271
60.8.37 FLEXCAN_WMB_CSn (Wake Up Message Buffer Register for C/S) ..... 1271
60.8.38 FLEXCAN_WMB_IDn (Wake Up Message Buffer Register for ID) ..... 1272
60.8.39 FLEXCAN_WMB_D03_n (Wake Up Message Buffer Register for Data 0-3) ..... 1273
60.8.40 FLEXCAN_WMB_D47_n (Wake Up Message Buffer Register Data 4-7) ..... 1273
60.8.41 FLEXCAN_EPRS (Enhanced CAN Bit Timing Prescalers Register) ..... 1274
60.8.42 FLEXCAN_ENCBT (Enhanced Nominal CAN Bit Timing Register) ..... 1275
60.8.43 FLEXCAN_EDCBT (Enhanced Data Phase CAN bit Timing Register) ..... 1276

Flagchip

60.8.44 FLEXCAN_ETDC (Enhanced Transceiver Delay Compensation Register) ..... 1277
60.8.45 FLEXCAN_FDCTRL (CAN FD Control Register) ..... 1278
60.8.46 FLEXCAN_FDCBT (CAN FD Bit Timing Register) ..... 1281
60.8.47 FLEXCAN_FDCRC (CAN FD CRC Register) ..... 1282
60.8.48 FLEXCAN_ERFCR (Enhanced Rx FIFO Control Register) ..... 1283
60.8.49 FLEXCAN_ERFIER (Enhanced Rx FIFO Interrupt Enable Register) ..... 1284
60.8.50 FLEXCAN_ERFSR (Enhanced Rx FIFO Status Register) ..... 1285
60.8.51 FLEXCAN_GATE_CS (Gateway Control Register) ..... 1287
60.8.52 FLEXCAN_GATE_BUFFERn (Gateway Buffer Register) ..... 1287
60.8.53 FLEXCAN_GATE_DUMMYn (Gateway Router Dummy Register) ..... 1288
60.8.54 FLEXCAN_ERX_FIFOn (Enhanced Rx FIFO Register) ..... 1288
60.8.55 FLEXCAN_ERFFELn (Enhanced Rx FIFO Filter Element Register) ..... 1289
**Chapter 61 Micro Second Channel (MSC) ..... 1290**
61.1 Revision History ..... 1290
61.2 Chip-specific Information ..... 1291
61.3 Features ..... 1291
61.4 Overview ..... 1291
61.5 Functional Description ..... 1292
61.5.1 Transmitting Channel ..... 1292
61.5.2 Receiving Channel ..... 1294
61.5.3 I/O Control ..... 1295
61.6 Application Information ..... 1295
61.7 Register Definition ..... 1295
61.7.1 MSC_GCR (Global Control Register) ..... 1296
61.7.2 MSC_CCULR (Channel Control Unlock Register) ..... 1297
61.7.3 MSC_RCCSR (Reception Channel Control and Status Register) ..... 1297
61.7.4 MSC_TCCTR (Transmitting Channel Control Register) ..... 1298
61.7.5 MSC_TCSTR (Transmitting Channel Status Register) ..... 1299
61.7.6 MSC_TCDAR (Transmitting Channel Data Register) ..... 1300
61.7.7 MSC_TCCOR (Transmitting Channel Command Register) ..... 1301

Flagchip

61.7.8 MSC_TCSLR (Transmitting Channel Source Selection Low Register) ..... 1301
61.7.9 MSC_TCSHR (Transmitting Channel Source Selection High Register) ..... 1303
61.7.10 MSC_TCELR (Transmitting Channel Emergency Load Register) ..... 1305
61.7.11 MSC_RDR0 (Reception Data Register0) ..... 1307
61.7.12 MSC_RDR1 (Reception Data Register1) ..... 1308
61.7.13 MSC_RDR2 (Reception Data Register2) ..... 1309
61.7.14 MSC_RDR3 (Reception Data Register3) ..... 1310
61.7.15 MSC_INCR (Interrupt Control Register) ..... 1311
61.7.16 MSC_INSR (Interrupt Status Register) ..... 1312
61.7.17 MSC_ISCR (Interrupt Set/Clear Register) ..... 1313
61.7.18 MSC_IOCR (Input/Output Control Register) ..... 1315
61.7.19 MSC_TCCTR1 (Transmitting Channel Control Register1) ..... 1316
61.7.20 MSC_RTOR (Reception Timeout Register) ..... 1317
61.7.21 MSC_SRCR (Software Reset Clear Register) ..... 1318
61.7.22 MSC_MSR (Software Reset Register) ..... 1318
**Chapter 62 Single Edge Nibble Transmission (SENT) ..... 1320**
62.1 Revision History ..... 1320
62.2 Chip-specific Information ..... 1321
62.3 Overview ..... 1321
62.3.1 Modes of Operation ..... 1321
62.4 Features ..... 1321
62.5 Block diagram ..... 1322
62.6 Functional Description ..... 1322
62.6.1 Message Definition ..... 1322
62.6.2 Standard Mode ..... 1324
62.6.3 SPC Mode ..... 1327
62.6.4 CRC Calculation ..... 1328
62.7 Initialization ..... 1329
62.7.1 Fast Message ..... 1329
62.7.2 Short Serial Message ..... 1329

Flagchip

62.7.3 Enhanced Serial Message ..... 1330
62.7.4 SPC MODE ..... 1330
62.7.5 Fast Message DMA Transfer ..... 1331
62.7.6 Fast Message DMA Transfer with FIFO Enable ..... 1331
62.7.7 Short/Enhanced Serial Message DMA Transfer ..... 1331
62.8 Register Definition ..... 1332
62.8.1 SENT_GLBL_CTL (Global Control Register) ..... 1333
62.8.2 SENT_CHN_EN (Channel Enable Register) ..... 1334
62.8.3 SENT_GLBL_STAT (Global Status Register) ..... 1335
62.8.4 SENT_FMSG_STAT (Fast Message Status Register) ..... 1335
62.8.5 SENT_SMSG_STAT (Slow Message Status Register) ..... 1336
62.8.6 SENT_FIFO_EN (FIFO Enable Register) ..... 1337
62.8.7 SENT_DATA_NUM_CTL1 (Each Channel Data Number Control Register) ..... 1337
62.8.8 SENT_OVFL_UDFL_STAT (Fast Message FIFO Overflow DMA Underflow State Register) ..... 1338
62.8.9 SENT_NB_MODE1 (Channel Nibble Data Mode Control 1 Register) ..... 1339
62.8.10 SENT_FDMA_CTL (Channel Fast Message DMA Control Register) ..... 1340
62.8.11 SENT_SDMA_CTL (Channel Slow Message DMA Control Register) ..... 1341
62.8.12 SENT_FINT_CTL (Channel Fast Message Interrupt Control Register) ..... 1341
62.8.13 SENT_SINT_CTL (Channel Slow Message Interrupt Control Register) ..... 1342
62.8.14 SENT_CHN_CLK_CTLn (Channel Clock Control Register) ..... 1342
62.8.15 SENT_CHN_STATn (Channel Status Register) ..... 1343
62.8.16 SENT_CHN_CTLn (Channel Control Register) ..... 1345
62.8.17 SENT_CHN_FDATAn (Channel Received Fast Data Nibble Value Register) ..... 1347
62.8.18 SENT_CHN_FCRCn (Channel Fast Message CRC and Status & Communication Value Register) . 1348
62.8.19 SENT_CHN_FTSn (Channel Fast Timestamp Value Register) ..... 1349
62.8.20 SENT_CHN_SBIT3_n (Channel Slow Message Status & Communication Bit 3 Received Value Register) 1349
62.8.21 SENT_CHN_SBIT2_n (Channel Slow Message Status & Communication Bit 2 Received Value Register) 1350
62.8.22 SENT_CHN_STSn (Channel Slow Message Timestamp Value Register) ..... 1351

Flagchip

62.8.23 SENT_CHN_DFDn (Channel n DMA Fast Data Register) ..... 1351
62.8.24 SENT_CHN_DFCn (Channel n DMA Fast CRC Data Register) ..... 1352
62.8.25 SENT_CHN_DFTSn (Channel n DMA Fast Timestamp Register) ..... 1352
62.8.26 SENT_CHN_DSB3_n (Channel n DMA Slow Message Bit3 Register) ..... 1353
62.8.27 SENT_CHN_DSB2_n (Channel n DMA Slow Message Bit2 Register) ..... 1353
62.8.28 SENT_CHN_DSTSn (Channel n DMA Slow Message Timestamp Register) ..... 1354
62.8.29 SENT_CHN_TSTMPn (Channel n Timestamp Register) ..... 1354
62.8.30 SENT_CHN_MCNTn (Channel n Message Counter Register) ..... 1355
62.8.31 SENT_CHN_SPC_CTLn (Channel n SPC Mode Control Register) ..... 1355
62.8.32 SENT_CHN_IDLE_CTLn (Channel n Idle Control Register) ..... 1356
**Chapter 63 FLASH API ..... 1358**
63.1 Revision History ..... 1358
63.2 Overview ..... 1359
63.3 Functional Description ..... 1359
63.3.1 Read ..... 1359
63.3.2 Program ..... 1359
63.3.3 Erase ..... 1359
63.3.4 Array Integrity Self Check ..... 1359
63.3.5 Margin Read Self Check ..... 1359
63.3.6 ECC Logic Check ..... 1359
63.3.7 EDC after ECC Logic Check ..... 1360
63.4 FLASH API Function ..... 1360
63.4.1 FlashRomApiVersion ..... 1362
63.4.2 FLASH_DRV_EraseBlock ..... 1362
63.4.3 FLASH_DRV_EraseBlock_Clear ..... 1364
63.4.4 FLASH_DRV_EraseSector ..... 1365
63.4.5 FLASH_DRV_EraseSector_Clear ..... 1366
63.4.6 FLASH_DRV_EraseNvr ..... 1367
63.4.7 FLASH_DRV_EraseNvr_Clear ..... 1367
63.4.8 FLASH_DRV_Program ..... 1368

Flagchip

63.4.9 FLASH_DRV_Program_Clear ..... 1370
63.4.10 FLASH_DRV_Program_Nvr ..... 1371
63.4.11 FLASH_DRV_Program_Nvr_Clear ..... 1372
63.4.12 FLASH_DRV_Array_Integrity_Check ..... 1372
63.4.13 FLASH_DRV_Margin_Read_Check ..... 1374
63.4.14 FLASH_DRV_Read_Status_Check ..... 1376
63.4.15 FLASH_DRV_Read_Status_Check_Clear ..... 1378
63.4.16 FLASH_DRV_HV_Status_Check ..... 1379
63.4.17 FLASH_DRV_Test_Status_Check ..... 1379
63.4.18 FLASH_DRV_Wdog_Configure ..... 1380
63.4.19 FLASH_DRV_Wdog_Status ..... 1381
63.4.20 FLASH_DRV_Wdog_Status_Clear ..... 1382
63.4.21 FLASH_DRV_EDC_Check ..... 1382
63.4.22 FLASH_DRV_ECC_Check ..... 1383
63.4.23 FLASH_DRV_Test_Mode_Close ..... 1384
63.4.24 FLASH_DRV_Read_Wait_CFG ..... 1384
63.4.25 FLASH_DRV_ENABLE_HOLD_CFG ..... 1386
63.4.26 FLASH_DRV_READ_ADDR_FUNC ..... 1386
63.4.27 FLASH_DRV_Test_Cfg_FUNC ..... 1387
63.4.28 FLASH_DRV_Array_Integrity_Suspend_FUNC ..... 1388
63.4.29 FLASH_DRV_Test_Mode_Open ..... 1389
63.4.30 FLASH_DRV_Sector_Program ..... 1389
63.4.31 FLASH_DRV_Read_Status_Check_128 ..... 1390
63.4.32 FLASH_DRV_Read_Status_Check_Clear_128 ..... 1390
63.4.33 FLASH_DRV_Array_Integrity_Check_128 ..... 1391
63.4.34 FLASH_DRV_Margin_Read_Check_128 ..... 1393
63.4.35 FLASH_DRV_CTL_CFG_FUNC ..... 1395
63.4.36 FLASH_DRV_READ_MASTER_ID_FUNC ..... 1396
63.4.37 FLASH_DRV_EDC_Check_128 ..... 1396
63.4.38 FLASH_DRV_ECC_Check_128 ..... 1397

Flagchip

63.4.39 FLASH_DRV_EAC_CHECK_128_HI ..... 1398
63.4.40 FLASH_DRV_AEC_CHECK ..... 1399
63.4.41 FLASH_DRV_MCTL_CFG_FUNC ..... 1399
63.4.42 FLASH_DRV_AMCTL_CFG_FUNC ..... 1401
63.4.43 FLASH_DRV_READ_PE_MASTER_ID_FUNC ..... 1402
63.4.44 FLASH_DRV_READ_PE_AMASTER_ID_FUNC ..... 1402
63.4.45 FLASH_DRV_EAC_CHECK_128_LO ..... 1403
63.4.46 FLASH_DRV_EDCA_CHECK_128_Hi ..... 1403
63.4.47 FLASH_DRV_EDCA_CHECK_128_LO ..... 1404
63.4.48 FLASH_DRV_ABORT_PE ..... 1404
63.4.49 FLASH_DRV_WRITE_ADR_FUNC ..... 1405

Flagchip

# List of Figures

Figure 1. FC7240F2M block diagram ..... 74
Figure 2. FC7240F2M system diagram ..... 79
Figure 3. Debug AP system ..... 103
Figure 4. Cortex-M7 debug structure ..... 104
Figure 5. System-level CPU debug architecture ..... 104
Figure 6. JTAG/SWD connection diagram ..... 105
Figure 7. State transition diagram of TAP Controller FSM ..... 106
Figure 8. SWJ-DP conceptual model ..... 107
Figure 9. JTAG and SWD switching FSM ..... 108
Figure 10. Debug AP and DP connection diagram ..... 109
Figure 11. CORDIC micro-operation diagram ..... 115
Figure 12. FC7240F2M top-level clocking diagram ..... 140
Figure 13. AON_CLK clock ..... 150
Figure 14. AON32K clock ..... 150
Figure 15. RTC 32K clock ..... 150
Figure 16. FLEXCAN PE clock ..... 151
Figure 17. ADC functional clock ..... 151
Figure 18. SENT functional clock ..... 152
Figure 19. MSC functional clock ..... 152
Figure 20. FTUx functional clock input ..... 153
Figure 21. FCPITx functional clock ..... 153
Figure 22. IP functional clock ..... 154
Figure 23. AONTIMER0 clock ..... 154
Figure 24. Reset sequence ..... 156
Figure 25. Boot flow ..... 157
Figure 26. FC7240F2M power diagram ..... 159
Figure 27. Overall FlexCore subsystem block diagram ..... 173

Flagchip

Figure 28. Safety interface between the FC7240 and SBC/PMIC ..... 177
Figure 29. Lockstep diagram ..... 195
Figure 30. Example of nested loop ..... 209
Figure 31. Memory array terms ..... 209
Figure 32. DMAMUX triggered channels ..... 214
Figure 33. DMAMUX channel triggering event ..... 215
Figure 34. WKU delay counter ..... 266
Figure 35. PMC block diagram ..... 272
Figure 36. CORDIC block diagram ..... 327
Figure 37. CORDIC application ..... 331
Figure 38. Mailbox block diagram ..... 452
Figure 39. Mailbox communication and interrupt channels ..... 453
Figure 40. FMC block diagram ..... 467
Figure 41. Read buffer ECC ..... 468
Figure 42. OTA diagram ..... 470
Figure 43. Overlay block diagram ..... 499
Figure 44. PCC block diagram ..... 559
Figure 45. WDOG block diagram ..... 622
Figure 46. EIM block diagram ..... 632
Figure 47. EIM Bus MUX4 diagram ..... 633
Figure 48. INTM interrupt active mode waveform ..... 663
Figure 49. INTM interrupt inactive mode waveform ..... 664
Figure 50. CMU block diagram ..... 671
Figure 51. Connection between TRGSEL3 and ISM ..... 680
Figure 52. ISM block diagram ..... 681
Figure 53. FPC logic ..... 681
Figure 54. No filter ..... 682
Figure 55. Immediate mode with filter ..... 682
Figure 56. Smart delay mode with timer decrement ..... 682
Figure 57. Smart delay mode with timer reset ..... 683

Flagchip

Figure 58. Prescaler mode ..... 683
Figure 59. LAM Logic ..... 684
Figure 60. FCSMU block diagram ..... 699
Figure 61. FCSMU configuration timeout ..... 706
Figure 62. FCSMU goes to fault state after warning timeout ..... 707
Figure 63. FCSMU clears fault in warning state ..... 707
Figure 64. FCSMU goes to fault state with interrupt output ..... 707
Figure 65. FCSMU goes to fault state with reset output ..... 707
Figure 66. Self-test structure ..... 734
Figure 67. FCSMU_PIN status when enabling both two pins ..... 735
Figure 68. FCSMU_PIN status when enabling only one pin ..... 735
Figure 69. TRGSEL block diagram ..... 782
Figure 70. LU block diagram ..... 786
Figure 71. ADC block diagram ..... 799
Figure 72. ADC ON/OFF control ..... 800
Figure 73. ADC start sequence ..... 802
Figure 74. ADC stop sequence ..... 803
Figure 75. ADC pre-trigger and trigger connection ..... 804
Figure 76. Single conversion mode by software trigger ..... 805
Figure 77. Single conversion mode by hardware trigger ..... 805
Figure 78. Continuous conversion mode by software trigger ..... 806
Figure 79. Continuous conversion mode by hardware trigger ..... 806
Figure 80. Sequence defined from SC_n register ..... 807
Figure 81. Sequence group mode ..... 808
Figure 82. CMP analog block diagram ..... 831
Figure 83. CMP digital signal flow ..... 833
Figure 84. TMU block diagram ..... 853
Figure 85. Filter bypass function ..... 855
Figure 86. FTU block diagram ..... 865
Figure 87. FTU counter up counting ..... 866

Flagchip

Figure 88. FTU counter up-down counting ..... 866
Figure 89. FTU counter free running counting ..... 866
Figure 90. Output compare application demo ..... 871
Figure 91. EPWM application demo ..... 872
Figure 92. CPWM0 application demo ..... 873
Figure 93. CPWM1 application demo ..... 873
Figure 94. Phase shift application demo ..... 874
Figure 95. Enhanced phase shift application demo ..... 875
Figure 96. Quad direction mode ..... 875
Figure 97. Quad phase mode ..... 876
Figure 98. MOD update FSM ..... 877
Figure 99. MOD update ..... 877
Figure 100. OUTMASK update ..... 878
Figure 101. Deadtime control ..... 880
Figure 102. Deadtime timeout ..... 880
Figure 103. Software output control ..... 881
Figure 104. Fault condition automatic clearing ..... 882
Figure 105. Fault condition manual clearing ..... 883
Figure 106. Channel match trigger ..... 883
Figure 107. Reload trigger ..... 884
Figure 108. Debug Output Control ..... 885
Figure 109. Debug Output Frozen ..... 885
Figure 110. GTB of FTU counter ..... 886
Figure 111. Channel Output Mask Control ..... 886
Figure 112. Complement Control ..... 887
Figure 113. FTU Counter interrupt generation ..... 887
Figure 114. Reload point interrupt generation ..... 887
Figure 115. Channel match interrupt generation ..... 888
Figure 116. Fault interrupt generation ..... 888
Figure 117. DMA request ..... 888

Flagchip

Figure 118. FCPIT block diagram ..... 922
Figure 119. PTIMER back-to-back operation in mode 0 ..... 963
Figure 120. PTIMER back-to-back operation in mode 2 ..... 964
Figure 121. PTIMER back-to-back operation in mode 4 ..... 964
Figure 122. FREQM block diagram ..... 982
Figure 123. TPU system diagram ..... 993
Figure 124. TPU block diagram ..... 994
Figure 125. Storage location of entry table ..... 996
Figure 126. TPU channel diagram ..... 1002
Figure 127. Channel mode logic and event flags ..... 1009
Figure 128. TCR1 clock selection ..... 1019
Figure 129. TCR2 filter ..... 1020
Figure 130. TCR2 prescaler ..... 1021
Figure 131. TBR_TPR register ..... 1024
Figure 132. TBR_T2R register ..... 1025
Figure 133. TBR_TRR register ..... 1026
Figure 134. EAC PLL ..... 1027
Figure 135. Angle counter system ..... 1027
Figure 136. FCSPI block diagram ..... 1086
Figure 137. Waveform of PCS-SCK configuration ..... 1088
Figure 138. FCIIC block diagram ..... 1105
Figure 139. FLEXCAN block diagram ..... 1175
Figure 140. MSC block diagram ..... 1291
Figure 141. Data repetition mode frame example ..... 1293
Figure 142. SENT block diagram ..... 1322
Figure 143. Encoding scheme for 6 data fast message ..... 1323
Figure 144. Example nominal nibble times ..... 1324
Figure 145. Construction of short serial data message from 16 SENT messages ..... 1325
Figure 146. Construction of enhanced serial data message from 18 SENT messages ..... 1326
Figure 147. Enhanced serial message format with 12-bit data field and 8-bit message ID ..... 1326

Flagchip

Figure 148. Enhanced serial message format with 16-bit data field and 4-bit message ID ..... 1327
Figure 149. SPC message format ..... 1327
Figure 150. Comparison of SAE CRC algorithm and Alt CRC algorithm ..... 1329
Figure 151. Flash API struct ..... 1360

Flagchip

# List of Tables

Table 1. FC7240F2M feature summary ..... 74
Table 2. Cortex-M7 configuration ..... 80
Table 3. System memory map ..... 81
Table 4. APB bridge 0 (AFCB0) slot assignments ..... 82
Table 5. APB bridge 1 (AFCB1) slot assignments ..... 86
Table 6. AHB memory map assignments ..... 91
Table 7. FC7240F2M PPB memory map ..... 91
Table 8. CM7_0 Interrupt vector assignments ..... 92
Table 9. WIC stop wake-up sources ..... 98
Table 10. DMAMUX assignments ..... 98
Table 11. JTAG/SWD pin configuration ..... 105
Table 12. IR code assignments ..... 107
Table 13. SYS AP register ..... 110
Table 14. SYS AP status register ..... 111
Table 15. SYS AP control register ..... 111
Table 16. Core Status0 register ..... 112
Table 17. Core Status1 register ..... 112
Table 18. SYS_AP Status1 register ..... 113
Table 19. WKU input channel connection ..... 115
Table 20. TRGSEL0 sources ..... 117
Table 21. TRGSEL1 sources ..... 120
Table 22. TRGSEL2 sources ..... 123
Table 23. TRGSEL3 sources ..... 123
Table 24. TRGSEL4 sources ..... 126
Table 25. TRGSEL5 sources ..... 129
Table 26. FREQM input sources ..... 134
Table 27. ADC0 Internal analog channels connection ..... 136

Flagchip

Table 28. ADC1 Internal analog channels connection ..... 136
Table 29. PCC0 part0 specification ..... 141
Table 30. PCC0 part1 specification ..... 145
Table 31. Reset sources ..... 155
Table 32. ISP communication instances and pins ..... 158
Table 33. FC7240F2M NVR configuration assignments ..... 162
Table 34. OTA content ..... 171
Table 35. Valid code for the lifecycle ..... 171
Table 36. FlexCore memory map ..... 173
Table 37. FlexCore AFCB ..... 174
Table 38. FlexCore interrupt assignments ..... 175
Table 39. TPU-related control ..... 176
Table 40. FCSMU channel assignments ..... 180
Table 41. EIM channel assignments ..... 181
Table 42. ERM channel assignments ..... 183
Table 43. EIM matrix error check (bank channels) ..... 183
Table 44. EIM matrix error check (lockstep channels) ..... 194
Table 45. Reset status of pins ..... 197
Table 46. Ports belonging to PD0 ..... 197
Table 47. Pad type ..... 197
Table 48. Modulo example ..... 213
Table 49. Un-align modulo example ..... 213
Table 50. WKU Wake-up sources ..... 265
Table 51. Translation between Flagchip MCU mode and ARM CPU mode ..... 280
Table 52. Available power modes ..... 280
Table 53. Differences in low-power mode behaviors for SRAM in standby mode ..... 281
Table 54. Triggers for state transition ..... 281
Table 55. Lifecycle status table ..... 304
Table 56. Lifecycle table ..... 304

Flagchip

Table 57. Difference between Secure Boot, Non-secure Boot HSM Part and Non-secure Boot Co-processor Part ..... 305
Table 58. MAM information in SOC ..... 436
Table 59. MAM0 master information ..... 436
Table 60. MAM0 slave information ..... 436
Table 61. MAM0 ACLRn registers information ..... 436
Table 62. MAM0 ACRn registers information ..... 437
Table 63. CRC checklist of SCG registers ..... 523
Table 64. Configuration CRC check covered registers ..... 704
Table 65. NVR data load of STCU ..... 736
Table 66. Channel decode ..... 801
Table 67. Configuring the trigger polarity ..... 803
Table 68. Configuring the trigger source ..... 803
Table 69. CMP connections ..... 830
Table 70. Available references ..... 830
Table 71. CMP function mode ..... 832
Table 72. FCPIT modes of operation ..... 921
Table 73. Pulse counter input options ..... 954
Table 74. PTIMER loop mode ..... 963
Table 75. Standard channel condition encoding scheme ..... 997
Table 76. Alternate channel condition encoding scheme ..... 999
Table 77. MSR signals - Match Service Requests ..... 1007
Table 78. TCAP and TSR signals - Transition Captures and Service Requests ..... 1007
Table 79. MCAP signal - Match Capture ..... 1007
Table 80. TBM2 signal - Transition Blocks Match 2 ..... 1007
Table 81. M1ET, M1EM2, M1BM2, M2BM1, M2BT signals ..... 1008
Table 82. Predefined Channel Mode control signals decoding ..... 1008
Table 83. Simultaneous match pin action priority ..... 1016
Table 84. Pulse widths and delays ..... 1018
Table 85. FCIIC modes of operation ..... 1106

Flagchip

Table 86. Clocks ..... 1144
Table 87. Resets ..... 1144
Table 88. Receiver wakeup options ..... 1146
Table 89. Break character length ..... 1148
Table 90. FLEXCAN I/O signals ..... 1175
Table 91. Composition of the arbitration value when local priority is disabled ..... 1180
Table 92. Composition of the arbitration value when local priority is disabled ..... 1180
Table 93. Mask mechanism ..... 1182
Table 94. Number of legacy Rx FIFO filters ..... 1183
Table 95. Standard ID filter element with filter + mask scheme (FSCH = b00) ..... 1193
Table 96. Standard ID filter element with range scheme (FSCH = b01) ..... 1193
Table 97. Standard ID filter element with two-filter scheme (FSCH = b10) ..... 1193
Table 98. Extended ID filter element with filter + mask scheme (FSCH = b00) ..... 1194
Table 99. Extended ID filter element with range scheme (FSCH = b01) ..... 1194
Table 100. Extended ID filter element with two-filter scheme (FSCH = b10) ..... 1194
Table 101. Time segment syntax ..... 1199
Table 102. Bosch CAN 2.0B standard compliant bit time segment settings ..... 1199
Table 103. FLEXCAN Registers in RAM ..... 1207
Table 104. Register access and reset information ..... 1208
Table 105. Registers supported in chip ..... 1209
Table 106. Message buffer structure example with 64 Bytes payload ..... 1211
Table 107. Message buffer code for Rx buffers ..... 1212
Table 108. Message buffer code for Tx buffers ..... 1214
Table 109. Data Bytes Validity ..... 1215
Table 110. 8-byte message buffer ..... 1216
Table 111. Legacy Rx FIFO data structure ..... 1217
Table 112. ID filter table structure ..... 1217
Table 113. Enhanced Rx FIFO data structure ..... 1218
Table 114. ID HIT and HR time stamp offsets ..... 1219
Table 115. ID filter element example ..... 1220

Flagchip

Table 116. SENT modes of operation ..... 1321

Table 117. Status and communication nibble description ..... 1324

Flagchip

# Chapter 1 About This Document

This reference manual provides complete information on the architecture, features, and peripherals of the Flagchip FC7240 microcontroller (MCU).

## 1.1 Audience

This manual targets system designers and software developers who are using the FC7240 MCU to develop their applications. The readers are expected to understand operating systems, logic circuits, microcontrollers, etc.

## 1.2 Abbreviations

The following abbreviations are used in this document.

| No. | Abbreviation | Description |
|---|---|---|
| 1 | ACC | ARMv7 Cache Controller |
| 2 | ADC | Analog-to-Digital Converter |
| 3 | AFCB | Advanced Flagchip Bus (APB Bridge) |
| 4 | AHB | Advanced High-Performance Bus |
| 5 | AHBD | AHB-Lite Debug |
| 6 | AHBP | AHB-Lite Peripheral |
| 7 | AHBS | AHB-Lite Slave |
| 8 | AMBA | Advanced Microcontroller Bus Architecture |
| 9 | AOI | AND/OR/INVERT |
| 10 | AONTIMER | Always-on Timer |
| 11 | AP | Access Port |
| 12 | APB | Advanced Peripheral Bus |
| 13 | AVB | Audio Video Bridging |
| 14 | AXI | Advanced eXtensible Interface |
| 15 | BEM | Bus Error Monitor |
| 16 | BG | Bandgap |
| 17 | BGA | Ball Grid Array |
| 18 | BIU | Bus Interface Unit |
| 19 | BJT | Bipolar Junction Transistor |
| 20 | BTB | Back-to-Back |
| 21 | CBC | Cipher Block Chaining |
| 22 | CCM | Counter with CBC-MAC |
| 23 | CCU | Code Cache Unit |
| 24 | CGC | Clock Gating Control |
| 25 | CIC | Cascaded Integrator Comb |
| 26 | CPM | CPU Private Module |
| 27 | CMAC | Cipher-based Message Authentication Code |
| 28 | CMU | Clock Monitor Unit |
| 29 | COCO | Conversion Complete |
| 30 | CPWM | Center-aligned Pulse-Width Modulation |

Flagchip

| No. | Abbreviation | Description |
|---|---|---|
| 31 | CRC | Cyclic Redundancy Check |
| 32 | CSC | CPU System Control |
| 33 | CTCM | Code Tightly-Coupled Memory |
| 34 | CTM | Cross Trigger Matrix |
| 35 | CTR | Counter |
| 36 | DAC | Digital-to-Analog Converter |
| 37 | DDR | Double Data Rate |
| 38 | DMA | Direct Memory Access |
| 39 | DMAC | DMA Controller |
| 40 | DMAMUX | Direct Memory Access Multiplexer |
| 41 | DMPU | Data Memory Protection Unit |
| 42 | DP | Debug Port |
| 43 | DSE | Driver Strength Enable |
| 44 | DSP | Digital Signal Processing |
| 45 | DTCM | Data Tightly-Coupled Memory |
| 46 | DWP | Domain Write Protection |
| 47 | DWT | Data Watchpoint and Trace |
| 48 | ECB | Electronic Codebook Book |
| 49 | ECC | Error Correction Code |
| 50 | ECM | Event Combiner Module |
| 51 | EEPROM | Electronically Erasable Programmable Read-Only Memory |
| 52 | EIM | Error Injection Module |
| 53 | EPPB | External Private Peripheral Bus |
| 54 | EPWM | Edge-aligned Pulse-Width Modulation |
| 55 | ERM | Error Reporting Module |
| 56 | ETM | Embedded Trace Macro |
| 57 | EVITA | E-safety Vehicle Intrusion Protected Applications |
| 58 | EVTG | Event Generator |
| 59 | FCIIC | Flagchip (FC) Inter-Integrated Circuit |
| 60 | FCPIT | Flagchip (FC) Programmable Interrupt Timer |
| 61 | FCSMU | Fault Control and Safety Management Unit |
| 62 | FCSPI | Flagchip (FC) Serial Peripheral Interface |
| 63 | FCUART | Flagchip (FC) Universal Asynchronous Receiver/Transmitter |
| 64 | FF | Flip-Flop |
| 65 | FIFO | First In First Out |
| 66 | FIRC | Fast Internal Reference Clock |
| 67 | FLEXCAN | FLEX Controller Area Network |
| 68 | FMC | Flash Memory Controller |
| 69 | FOSC | Fast Oscillator |
| 70 | FPB | Flash Patch and Breakpoint |
| 71 | FPC | Filter & Prescaler Channel |

Flagchip

| No. | Abbreviation | Description |
|---|---|---|
| 72 | FPM | Full Performance Mode |
| 73 | FPU | Floating Point Unit |
| 74 | FREQM | Frequency Measurement |
| 75 | FSE | Fast Select Enable |
| 76 | FSM | Finite State Machine |
| 77 | FTU | Flexible Timer Unit |
| 78 | GCM | Galois/Counter Mode |
| 79 | GPIO | General-Purpose Input/Output |
| 80 | GTB | Global Time Base |
| 81 | HMI | Human-Machine Interface |
| 82 | HSM | Hardware Secure Module |
| 83 | HVD | High-Voltage Detect |
| 84 | I2S | Inter-IC Sound |
| 85 | IIC/I2C | Inter-Integrated Circuit |
| 86 | IMPU | Instruction Memory Protection Unit |
| 87 | INTM | Interrupt Monitor |
| 88 | INTU | Interrupt Unit |
| 89 | IRC | Internal Reference Clock |
| 90 | IRQ | Interrupt Request |
| 91 | ISM | Interface Safety Monitor |
| 92 | ISP | In-System Program |
| 93 | ITCM | Instruction Tightly-Coupled Memory |
| 94 | ITM | Instrumentation Trace Macrocell |
| 95 | LAM | Logic Analyzer Module |
| 96 | LBIST | Logic Built-In Self-Test |
| 97 | LDO | Low Dropout |
| 98 | LIN | Local Interconnect Network |
| 99 | LQFP | Low-profile Quad Flat Package |
| 100 | LU | Logic Unit |
| 101 | LVD | Low-Voltage Detect |
| 102 | LVDS | Low Voltage Differential Signaling |
| 103 | LVR | Low-Voltage Reset |
| 104 | MAM | Matrix Access Monitor |
| 105 | MAP | Memory Access Protection |
| 106 | MB | Mailbox |
| 107 | MBIST | Memory Built-In Self-Test |
| 108 | MDC | Management Data Clock |
| 109 | MDI | Mailbox Data Input |
| 110 | MDIO | Management Data Input/Output |
| 111 | MISO | Master In Slave Out |
| 112 | MOSI | Master Out Slave In |

Flagchip

| No. | Abbreviation | Description |
|---|---|---|
| 113 | MPU | Memory Protection Unit |
| 114 | MSC | Micro Second Channel |
| 115 | MVA | Modified Virtual Address |
| 116 | NMI | Non-maskable Interrupt |
| 117 | NPN | Negative-Positive-Negative |
| 118 | NVIC | Nested Vectored Interrupt Controller |
| 119 | NVM | Non-volatile Memory |
| 120 | NVR | Non-volatile Region |
| 121 | OPAM | Operational Amplifier |
| 122 | OSC | Oscillator |
| 123 | PAP | Peripheral Access Protection |
| 124 | PCC | Peripheral Configuration Control |
| 125 | PIT | Programmable Interrupt Timer |
| 126 | PLL | Phase-Locked Loop |
| 127 | PMC | Power Management Controller |
| 128 | PMIC | Power Management Integrated Circuit |
| 129 | PNET | Pretended Networking |
| 130 | POR | Power-on Reset |
| 131 | PPB | Private Peripheral Bus |
| 132 | PTAT | Proportional-to-Absolute Temperature |
| 133 | PTP | Precision Time Protocol |
| 134 | PWM | Pulse Width Modulation |
| 135 | RCC | Reset and Clock Controller |
| 136 | RGM | Reset Generation Module |
| 137 | RPM | Reduce Power Mode |
| 138 | RSA | Rivest–Shamir–Adleman |
| 139 | RTC | Real-Time Clock |
| 140 | RWDS | Read/Write Data Strobe |
| 141 | RWW | Read-While-Write |
| 142 | SBC | System Basis Chip |
| 143 | SCG | System Clock Generator |
| 144 | SCL | Serial Clock Line |
| 145 | SCM | System Control Module |
| 146 | SCS | System Control Space |
| 147 | SDA | Serial Data Line |
| 148 | SDMA | Simple Direct Memory Access |
| 149 | SDR | Single Data Rate |
| 150 | SEC | Security Controller |
| 151 | SECDED | Single-bit Error Correction and Double-bit Error Detection |
| 152 | SEMA | Semaphore |
| 153 | SENT | Single Edge Nibble Transmission |

| No. | Abbreviation | Description |
|---|---|---|
| 154 | SIMD | Single Instruction Multiple Data |
| 155 | SIRC | Slow Internal Reference Clock |
| 156 | SMBUS | System Management Bus |
| 157 | SMC | System Mode Controller |
| 158 | SOG | Sea of Gate |
| 159 | SOSC | Slow Oscillator |
| 160 | SPC | Short PWM Code |
| 161 | SPI | Serial Peripheral Interface |
| 162 | SRE | Slew Rate Enable |
| 163 | STCU | Safety Test and Control Unit |
| 164 | SWD | Serial Wire Debug |
| 165 | SWJ-DP | Serial Wire/JTAG Debug Port |
| 166 | SysTick | System Tick Timer |
| 167 | TAP | Test Access Port |
| 168 | TCM | Tightly-Coupled Memory |
| 169 | TMU | Temperature Monitor Unit |
| 170 | TPIU | Trace Port Interface Unit |
| 171 | TPU | Time Processing Unit |
| 172 | TRGSEL | Trigger Select |
| 173 | TSN | Time-Sensitive Networking |
| 174 | TSTMP | Timer Stamp |
| 175 | UART | Universal Asynchronous Receiver and Transmitter |
| 176 | VCO | Voltage-Controlled Oscillator |
| 177 | WDOG | Watchdog |
| 178 | WIC | Wake-up Interrupt Controller |
| 179 | WKU | Wake-up Unit |

Flagchip

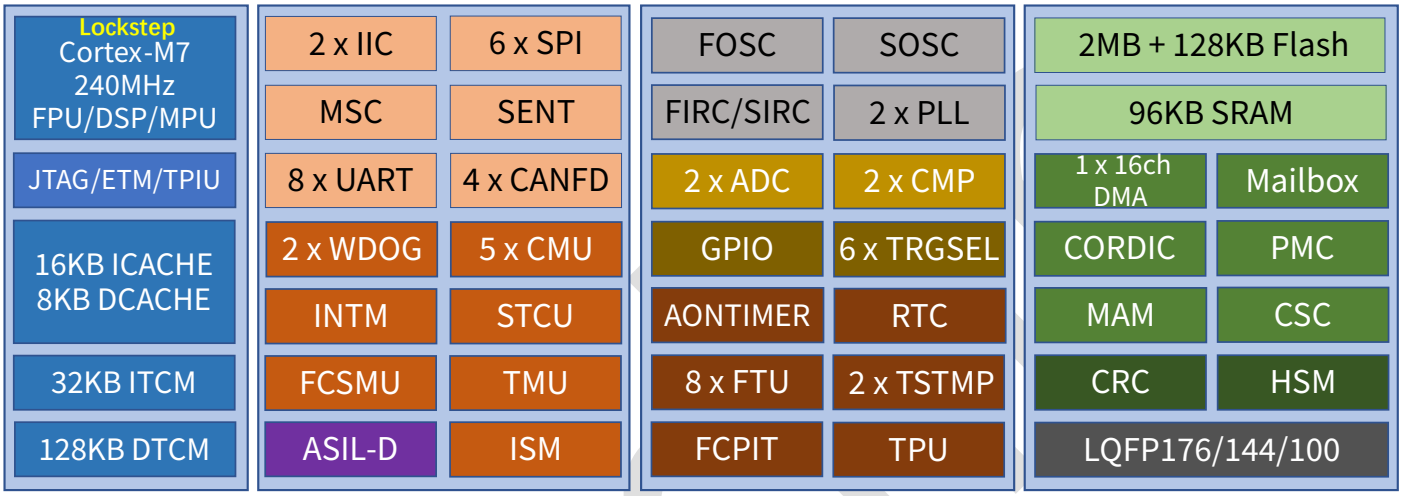
# Chapter 2 Features

This section summarizes the FC7240F2M features.

## 2.1 Block Diagram

The following block diagram lists the major features of the FC7240F2M.

**Figure 1. FC7240F2M block diagram**




## 2.2 Feature Summary

**Table 1. FC7240F2M feature summary**

| Feature | FC7240F2M |
|---|---|
| **Hardware Characteristics** | |
| Package | • 176-pin LQFP, 24*24 mm, 0.5 mm pitch, exposed pad<br>• 144-pin LQFP, 16*16 mm, 0.4mm pitch, exposed pad<br>• 100-pin LQFP, 12*12 mm, 0.4mm pitch, exposed pad |
| Voltage range | • 3.0 V to 5.5 V |
| Temperature range (Ta) | • -40°C to 125°C |
| Temperature range (Tj) | • -40°C to 150°C |
| **System** | |
| CPU | • 1 × ARM Cortex M7 Run Core, DSP/Single precision FPU support, extra one Check Core to form lock step for CPU0<br>• MPU enabled with 16 regions<br>• 16 KB I-Cache, 8KB D-Cache with ECC<br>• 32 KB ITCM, 64 KB DTCM0 and 64 KB DTCM1 with ECC for each run Core<br>• Overlay support |
| Max. CPU frequency | • 240 MHz |
| Max. Bus frequency | • 120 MHz |
| Max. Slow frequency | • 60 MHz |
| Max. Flash read frequency | • 33.3 MHz |
| Wake-up Unit (WKU) | • Yes<br>• Internal peripheral as wakeup source |

**Table 1. FC7240F2M feature summary (continued)**

| Feature | FC7240F2M |
|---|---|
| Direct Memory Access (DMA) | • 1 × 16-channel DMA with lockstep |
| DMA Multiplexer (DMAMUX) | • 1 × 128 DMA requests |
| Non-maskable Interrupt (NMI) | • Yes |
| Debug & Trace | • 4-pin serial wire debug (SWD/JTAG) + TPIU + ETM |
| Power Management Controller (PMC)/ System Mode Controller (SMC)/ Reset Generation Module (RGM) | • Low voltage warning and detect with selectable trip points<br>• Single supply input; support 1.5 V, 1.1 V and 2.5 V supply generation<br>• Support 1.5 V supply input to generate 1.1 V<br>• Power Mode Control<br>• Reset Control<br>• Reset watchdog |
| Matrix (MAM) | • 1 × Matrix with Matrix Access Monitor (access permission check and watchdog monitor) |
| APB Bridge (AFCB) | • 2 × APB Bridges |
| Security Controller (SEC) | • Security Control Module |
| CORDIC | • CORDIC Accelerator with scale |
| CPU System Control (CSC) | • 1 × CPU System Control |
| System Control Module (SCM) | • Support the master and slave access ECC monitor control<br>• Support the System Miscellaneous Control |
| Mail Box (MB) | • Support multi-core communication<br>• Support SEMA function |
| **Memory** | |
| Flash Controller (FC) | • 2 MB Program Flash (PFlash) (if not parallel read mode, support SWP/OTA)<br>• 128 KB Data Flash (DFlash) for Emulate EEPROM<br>• ECC support |
| Flash Memory Controller (FMC) | • 1 × FMC with two read prefetch and buffers |
| RAM | • 64 KB + 32KB SRAM slot<br>• ECC with double/single bit error report<br>• Support address and data ECC |
| ROM | • 128 KB ROM with ECC |
| **Clocks (SCG)** | |
| External crystal oscillator or resonator | • SOSC: 32 - 40 kHz<br>• FOSC: 16 - 48 MHz |
| External square wave input clock | • DC to 50 MHz |
| Internal clock references | • 96 MHz FIRC<br>• 12 MHz SIRC<br>• 32 kHz SIRC |
| Phase-Locked Loop (PLL) | • 2 × PLLs<br>• 2 – 4 MHz reference clock (after divide) |

**Table 1. FC7240F2M feature summary (continued)**

| Feature | FC7240F2M |
|---|---|
| **Human-Machine Interface (HMI)** | |
| General Purpose Input/Output (GPIO) | • 5.5 V I/O<br>• Support 5.5V segment<br>• Default to disabled<br>• Hysteresis and configurable pull and pull direction on all input pins<br>• Configurable slew rate on normal pad and drive strength on fast pads |
| Port Control (PORT) | • Port Control Module |
| Trigger Select (TRGSEL) | • 6 × TRGSELs |
| Logic Unit (LU) | • 1 × Logic Unit |
| **Analog** | |
| Analog-to-Digital Converter (ADC) | • 2 × ADCs<br>• 12-bit resolution<br>• Differential or single channel |
| Comparator with internal Digital-to-Analog Converter (DAC) | • 2 × CMPs<br>• 8-bit DAC<br>• Up to 8 input channels |
| Temperature Monitor Unit (TMU) | • Yes. Support both PTAT for ADC and two temperature monitors (125°C/150°C warning) |
| 16-bit Flexible Timer Unit (FTU) | • 8 × FTUs with 8 channels<br>• Support Basic Input Capture/Output Compare function<br>• FTU1/2 with Quadrature Decoder, 24-bit count<br>• Support dead time function<br>• Support enhanced input capture function<br>• DMA support |
| FC Programmable Interrupt Timer (FCPIT) | • 1 × FCPIT<br>• Each FCPIT supports 4 channels<br>• DMA support |
| Timer Stamp (TSTMP) | • 2 × TSTMP timers<br>• TSTMP0 running at 1MHz divide from SIRC12M<br>• TSTMP1 running from Bus clock<br>• Support 4 channels interrupt generation on time reach |
| Real-Time Clock (RTC) | • Hardware compensation |
| Always-on Timer (AONTIMER) | • 1-channel<br>• 32-bit pulse counter or periodic interrupt<br>• Functional in all power modes |
| Programmable Timer (PTIMER) | • 2 × PTIMERs |
| Frequency Measurement (FREQM) | • Support to measure the multi modules clock output |

Flagchip

**Table 1. FC7240F2M feature summary (continued)**

| Feature | FC7240F2M |
|---|---|
| Time Processing Unit (TPU) | • Up to 32 channels<br>• Support different input capture and output functions<br>• Support Angel Count mode |
| **Communication Interfaces** | |
| FC Serial Peripheral Interface (FCSPI) | • 6 × FCSPIs<br>• 8 TX/RX FIFOs<br>• Master mode and slave mode functions<br>• Supports multiple chip selects in master mode<br>• Support 1-4bit data transfer.<br>• DMA Support |
| FC Inter-Integrated Circuit (FCIIC) | • 2 × Standard SMBUS compatible IIC<br>• 8-word FIFO support<br>• DMA support |
| FC Universal Asynchronous Receiver/Transmitter (FCUART) | • 8 × FCUARTs<br>• Supports LIN protocol versions 1.3, 2.0, 2.1, 2.2A, and SAE J2602<br>• Standard features<br>• LIN slave operation support<br>• Support DMA<br>• 16-word FIFO |
| FLEX Controller Area Network (FLEXCAN) | • 4 × FLEXCANs with optional ISO CAN-FD<br>• Support 32 Message Buffers with ECC check<br>• PNET feature support for FLEXCAN0/1/2/3<br>• DMA support |
| Single Edge Nibble Transmission (SENT) | • 1 × SENT<br>• Each SENT supports 4 fast and slow channels |
| Micro Second Channel (MSC) | • 1 × MSC without LVDS<br>• Support both up stream and down stream<br>• Up to 25 MHz on normal pad |
| **Security** | |
| Cyclic Redundancy Check (CRC) | • 1 × programmable CRC |
| Hardware Secure Module (HSM) | • Dedicated Core for HSM module or FlexCore mode |
| | • Standalone asymmetric accelerator, such as RSA/ECC/SM2/SM9 |
| | • Standalone symmetric accelerator<br>• AES/SM4, encryption and decryption<br>• Support ECB, CBC, CTR, CCM, GCM, and CMAC modes |
| | • Standalone hash accelerator<br>• Support SM3/SHA etc. |
| | • Two Random Number Generators |
| | • Key management, including key import/export |

**Table 1. FC7240F2M feature summary (continued)**

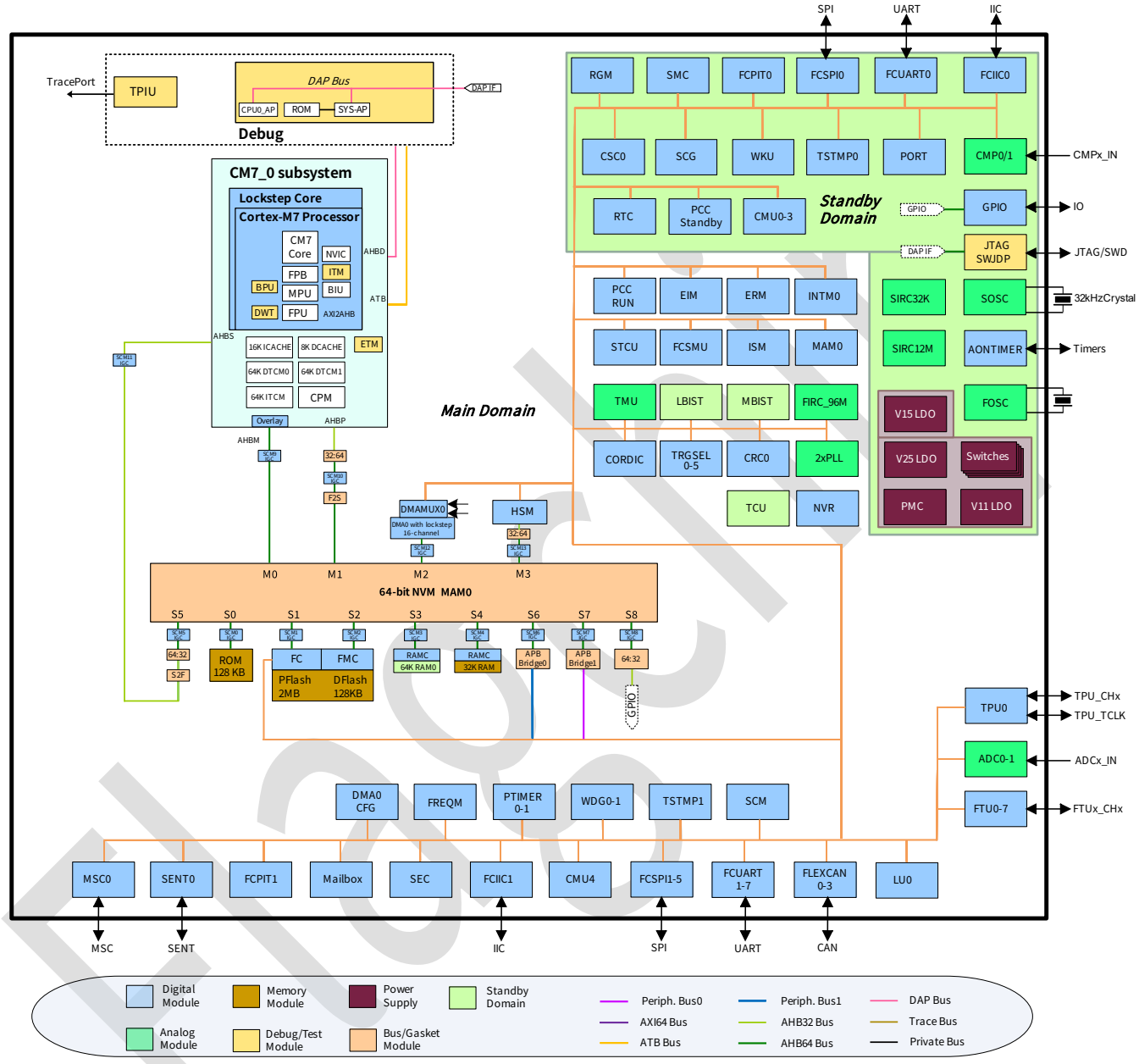
| Feature | FC7240F2M |
|---|---|
| **Safety** | |
| Watchdog (WDOG) | • 2 × WDOGs<br>• Independent clock sources |
| Fault Control and Safety Management Unit (FCSMU) | • Yes |
| Error Injection/Monitor | • EIM/ERM |
| Interrupt Monitor (INTM) | • 1 × INTM |
| Clock Monitor Unit (CMU) | • 5 × CMUs |
| Interface Safety Monitor (ISM) | • Yes. Support up to 8 reference/monitors |
| Safety Test and Control Unit (STCU) | • Yes. Support the LBIST/MBIST Control |
| **OTA** | |
| Flash Swap | • Yes. For 64-bit (non-parallel) mode configuration |

Flagchip

## 2.3 System Diagram

The system diagram FC7240F2M is shown below.

**Figure 2. FC7240F2M system diagram**



# Chapter 3 Cortex-M7 Configuration

The table below shows the Cortex-M7 (CM7) configuration. CM7 3/4 are lock step checker cores, which work with CM7 0/1 together.

**Table 2. Cortex-M7 configuration**

| Description | CM7 configuration |
|---|---|
| Floating point unit (FPU) configuration | Single-precision FPU |
| DSP Extension instructions are included | Single cycle 16/32-bit MAC Single cycle dual 16-bit MAC |
| | 8/16-bit SIMD arithmetic |
| | Hardware Divide (2-12 Cycles) |
| I-cache configuration | I-cache implemented |
| D-cache configuration | D-cache implemented |
| Caches ECC configuration | Caches ECC implemented |
| MPU region configuration | 16 regions |
| Number of IRQs | 91 |
| IRQ priority width configuration | 3 (8 interrupt priority levels) |
| Debug (breakpoint/watchpoint) configuration | Full comparator set: 4 DWT and 8 FPB comparators |
| Internal Trace support | ITM and DWT trace functionality implemented |
| ETM support | ETM interface |
| WIC support | Yes |
| Number of WIC lines | 94 |
| Reset-all-registers | All asynchronously reset |
| I-cache size | 16 KB |
| D-cache size | 8 KB |

Flagchip

# Chapter 4 Memory Map

This chapter summarizes the FC7240 memory map. It includes a peripheral bridge, interrupt vectors, and DMA mux assignments to respective source modules.

## 4.1 System Memory Map

The FC7240F2M has 2 MB program flash (PFlash), 128 KB data flash (DFlash), 96 KB SRAM, and 1 × 160 KB TCM (32 KB ITCM and 128 KB DTCM). For the detailed memory map, refer to the below table.

**Table 3. System memory map**

| Start Address (hex) | End Address (hex) | Approx. Size | Comments | FC7240F2M |
|---|---|---|---|---|
| 0000_0000 | 0003_FFFF | 256 KB | ITCMx/ITCM Backdoor Address | 32 KB<br>***Note:*** FlexCore/HSM is not able to access this region. |
| 0100_0000 | 011F_FFFF | 2048 KB | PFlash for PORT0 | 2048 KB for each PORT |
| 0120_0000 | 013F_FFFF | 2048 KB | Reserved | |
| 0180_0000 | 027F_FFFF | 4*4096 KB | Reserved | Reserved Program Flash for PORTx |
| 0400_0000 | 0403_FFFF | 256 KB | DFlash | 128 KB |
| 0440_0000 | 0441_FFFF | 128 KB | NVR | 40 KB<br>NVR0: 0x0440_0000 to 0x0440_3FFF<br>NVR1: 0x0440_8000 to 0x0440_BFFF<br>NVR2: 0x0441_0000 to 0x0441_1FFF |
| 0442_0000 | 0442_1FFF | 8 KB | Reserved | |
| 0480_0000 | 0481_FFFF | 128 KB | ROM | |
| 0800_0000 | 080F_FFFF | 1024 KB | FAR region 0 | For Core only |
| 0820_0000 | 083F_FFFF | 2048 KB | Reserved | |
| 1000_0000 | 1FFF_FFFF | 256 MB | Reserved | |
| 2000_0000 | 2007_FFFF | 512 KB | DTCMx/DTCM Backdoor Address | 128 KB |
| 2100_0000 | 2100_FFFF | 64 KB | SRAM_0 | 64 KB |
| 2101_0000 | 2101_7FFF | 32 KB | SRAM_1 | 32 KB |
| 2104_0000 | 2107_FFFF | 256 KB | Reserved | |
| 2200_0000 | 2201_FFFF | 128 KB | Reserved | |
| 2210_0000 | 2211_FFFF | 128 KB | Reserved | |
| 2220_0000 | 2221_FFFF | 128 KB | Reserved | |
| 2230_0000 | 2251_FFFF | 3*128 KB | Reserved | |
| 2400_0000 | 2400_FFFF | 64 KB | Reserved | |
| 2410_0000 | 2410_FFFF | 64 KB | Reserved | |
| 2420_0000 | 2420_FFFF | 64 KB | Reserved | |
| 2430_0000 | 2450_FFFF | 3*64 KB | Reserved | |
| 4000_0000 | 401F_FFFF | 2048 KB | AFCB0 | |
| 4040_0000 | 405F_FFFF | 2048 KB | AFCB1 | |

Flagchip

**Table 3. System memory map (continued)**

| Start Address (hex) | End Address (hex) | Approx. Size | Comments | FC7240F2M |
|---|---|---|---|---|
| 4100_0000 | 41FF_FFFF | 16 MB | AHB ports/RGPIO | 64 KB/per AHB port |
| 4200_0000 | 5FFF_FFFF | 480 MB | Reserved | |
| 6700_0000 | 67FF_FFFF | 128 MB | Reserved | |
| 6800_0000 | 6FFF_FFFF | 128 MB | Reserved | |
| E000_0000 | E00F_FFFF | 1024 KB | Private Peripheral Bus | |

***Note:*** To support secure boot, the NVR_0/1 are used for HSM firmware and NVR2 are used for HSM key storage.

For FC7240F2M, the Overlay function supports up to 3 overlay regions. Each overlay region size can be from 1KB to 64KB with 1KB step size. For the Flash Alias Region, the region size can be from 64KB to 1MB with 64KB step size.

## 4.2 APB Bridge Assignments

The FC7240F2M has two APB bridges: AFCB0 and AFCB1.

### 4.2.1 AFCB0 Slot Assignments

**Table 4. APB bridge 0 (AFCB0) slot assignments**

| Peripheral Description | Peripheral Instance | SEG0 Slot | SEG1 Slot | Size (KB) | Start Address (hex) | End Address (hex) |
|---|---|---|---|---|---|---|
| Reserved | Reserved | 0 | | 4 | 4000_0000 | 4000_0FFF |
| Reserved | Reserved | 1 | | 4 | 4000_1000 | 4000_1FFF |
| Reserved | Reserved | 2 | | 4 | 4000_2000 | 4000_2FFF |
| Reserved | Reserved | 3 | | 4 | 4000_3000 | 4000_3FFF |
| Reserved | Reserved | 4 | | 4 | 4000_4000_ | 4000_4FFF |
| Reserved | Reserved | 5 | | 4 | 4000_5000 | 4000_5FFF |
| Reserved | Reserved | 6 | | 4 | 4000_6000 | 4000_6FFF |
| Reserved | Reserved | 7 | | 4 | 4000_7000 | 4000_7FFF |
| DMA Controller | DMA0 | 8 | | 4 | 4000_8000 | 4000_8FFF |
| DMA Configuration | DMACFG0 | 9 | | 4 | 4000_9000 | 4000_9FFF |
| DMA Mux | DMAMUX0 | 10 | | 4 | 4000_A000 | 4000_AFFF |
| Reserved | Reserved | 11 | | 4 | 4000_B000 | 4000_BFFF |
| Reserved | Reserved | 12 | | 4 | 4000_C000 | 4000_CFFF |
| Reserved | Reserved | 13 | | 4 | 4000_D000 | 4000_DFFF |
| Reserved | Reserved | 14 | | 4 | 4000_E000 | 4000_EFFF |
| Fault Control and Safety Management Unit | FCSMU | 15 | | 4 | 4000_F000 | 4000_FFFF |
| Reserved | Reserved | 16 | | 4 | 4001_0000 | 4001_0FFF |
| Reserved | Reserved | 17 | | 4 | 4001_1000 | 4001_1FFF |
| Reserved | Reserved | 18 | | 4 | 4001_2000 | 4001_2FFF |
| ROM | ROM | 19 | | 4 | 4001_3000 | 4001_3FFF |
| Security Controller | SEC | 20 | | 4 | 4001_4000 | 4001_4FFF |
| Reserved | Reserved | 21 | | 4 | 4001_5000 | 4001_5FFF |

**Table 4. APB bridge 0 (AFCB0) slot assignments (continued)**

| Peripheral Description | Peripheral Instance | SEG0 Slot | SEG1 Slot | Size (KB) | Start Address (hex) | End Address (hex) |
|---|---|---|---|---|---|---|
| Reserved | Reserved | 22 | | 4 | 4001_6000 | 4001_6FFF |
| Reserved | Reserved | 23 | | 4 | 4001_7000 | 4001_7FFF |
| Error Reporting Module | ERM | 24 | | 4 | 4001_8000 | 4001_8FFF |
| Error Injection Module | EIM | 25 | | 4 | 4001_9000 | 4001_9FFF |
| Interrupt Monitor | INTM0 | 26 | | 4 | 4001_A000 | 4001_AFFF |
| Interface Safety Monitor | ISM0 | 27 | | 4 | 4001_B000 | 4001_BFFF |
| Reserved | Reserved | 28 | | 4 | 4001_C000 | 4001_CFFF |
| Reserved | Reserved | 29 | | 4 | 4001_D000 | 4001_DFFF |
| Flash Memory Controller | FMC0 | 30 | | 4 | 4001_E000 | 4001_EFFF |
| FMC HSM | FMC_HSM | 31 | | 4 | 4001_F000 | 4001_FFFF |
| Flash Controller | FC | | 0 | 4 | 4002_0000 | 4002_0FFF |
| Flash Controller ALT | FCA | | 1 | 4 | 4002_1000 | 4002_1FFF |
| Software watchdog | WDOG0 | | 2 | 4 | 4002_2000 | 4002_2FFF |
| System Clock Generator | SCG | | 3 | 4 | 4002_3000 | 4002_3FFF |
| Peripheral Configuration Control | PCC0 | | 4 | 4 | 4002_4000 | 4002_4FFF |
| Reserved | Reserved | | 5 | 4 | 4002_5000 | 4002_5FFF |
| Trigger Multiplexing Selection | TRGSEL0 | | 6 | 4 | 4002_6000 | 4002_6FFF |
| | TRGSEL1 | | 7 | 4 | 4002_7000 | 4002_7FFF |
| | TRGSEL2 | | 8 | 4 | 4002_8000 | 4002_8FFF |
| | TRGSEL3 | | 9 | 4 | 4002_9000 | 4002_9FFF |
| Cyclic Redundancy Check | CRC0 | | 10 | 4 | 4002_A000 | 4002_AFFF |
| CORDIC Accelerator | CORDIC0 | | 11 | 4 | 4002_B000 | 4002_BFFF |
| Timer Stamp | TSTMP0 | | 12 | 4 | 4002_C000 | 4002_CFFF |
| | TSTMP1 | | 13 | 4 | 4002_D000 | 4002_DFFF |
| FC Programmable Interrupt Timer | FCPIT0 | | 14 | 4 | 4002_E000 | 4002_EFFF |
| Always-on Timer | AONTIMER0 | | 15 | 4 | 4002_F000 | 4002_FFFF |
| Real-Time Clock | RTC | | 16 | 4 | 4003_0000 | 4003_0FFF |
| Clock Monitor Unit | CMU0 | | 17 | 4 | 4003_1000 | 4003_1FFF |
| | CMU1 | | 18 | 4 | 4003_2000 | 4003_2FFF |
| | CMU2 | | 19 | 4 | 4003_3000 | 4003_3FFF |
| | CMU3 | | 20 | 4 | 4003_4000 | 4003_4FFF |
| | CMU4 | | 21 | 4 | 4003_5000 | 4003_5FFF |
| Reserved | Reserved | | 22 | 4 | 4003_6000 | 4003_6FFF |
| Programmable Timer | PTIMER0 | | 23 | 4 | 4003_7000 | 4003_7FFF |
| | PTIMER1 | | 24 | 4 | 4003_8000 | 4003_8FFF |
| Reserved | Reserved | | 25 | 4 | 4003_9000 | 4003_9FFF |

**Table 4. APB bridge 0 (AFCB0) slot assignments (continued)**

| Peripheral Description | Peripheral Instance | SEG0 Slot | SEG1 Slot | Size (KB) | Start Address (hex) | End Address (hex) |
|---|---|---|---|---|---|---|
| Reserved | Reserved | | 26 | 4 | 4003_A000 | 4003_AFFF |
| Analog-to-Digital Converter | ADC0 | | 27 | 4 | 4003_B000 | 4003_BFFF |
| | ADC1 | | 28 | 4 | 4003_C000 | 4003_CFFF |
| Reserved | Reserved | | 29 | 4 | 4003_D000 | 4003_DFFF |
| Reserved | Reserved | | 30 | 4 | 4003_E000 | 4003_EFFF |
| Wake-up Unit | WKU | | 31 | 4 | 4003_F000 | 4003_FFFF |
| Analog Comparator | CMP0 | | 32 | 4 | 4004_0000 | 4004_0FFF |
| | CMP1 | | 33 | 4 | 4004_1000 | 4004_1FFF |
| Reserved | Reserved | | 34 | 4 | 4004_2000 | 4004_2FFF |
| Temperature Monitor Unit | TMU | | 35 | 4 | 4004_3000 | 4004_3FFF |
| Power Management Controller | PMC | | 36 | 4 | 4004_4000 | 4004_4FFF |
| System Mode Controller | SMC | | 37 | 4 | 4004_5000 | 4004_5FFF |
| Reset Generation Module | RGM | | 38 | 4 | 4004_6000 | 4004_6FFF |
| Port A control | Port A | | 39 | 4 | 4004_7000 | 4004_7FFF |
| Port B control | Port B | | 40 | 4 | 4004_8000 | 4004_8FFF |
| Port C control | Port C | | 41 | 4 | 4004_9000 | 4004_9FFF |
| Port D control | Port D | | 42 | 4 | 4004_A000 | 4004_AFFF |
| Port E control | Port E | | 43 | 4 | 4004_B000 | 4004_BFFF |
| Reserved | Reserved | | 44 | 4 | 4004_C000 | 4004_CFFF |
| Reserved | Reserved | | 45 | 4 | 4004_D000 | 4004_DFFF |
| Reserved | Reserved | | 46 | 4 | 4004_E000 | 4004_EFFF |
| Reserved | Reserved | | 47 | 4 | 4004_F000 | 4004_FFFF |
| Reserved | Reserved | | 48 | 4 | 4005_0000 | 4005_0FFF |
| Reserved | Reserved | | 49 | 4 | 4005_1000 | 4005_1FFF |
| Reserved | Reserved | | 50 | 4 | 4005_2000 | 4005_2FFF |
| Reserved | Reserved | | 51 | 4 | 4005_3000 | 4005_3FFF |
| Single Edge Nibble Transmission | SENT0 | | 52 | 4 | 4005_4000 | 4005_4FFF |
| Reserved | Reserved | | 53 | 4 | 4005_5000 | 4005_5FFF |
| Reserved | Reserved | | 54 | 4 | 4005_6000 | 4005_6FFF |
| Reserved | Reserved | | 55 | 4 | 4005_7000 | 4005_7FFF |
| Mailbox | MB | | 56 | 4 | 4005_8000 | 4005_8FFF |
| Reserved | Reserved | | 57 | 4 | 4005_9000 | 4005_9FFF |
| Reserved | Reserved | | 58 | 4 | 4005_A000 | 4005_AFFF |
| Reserved | Reserved | | 59 | 4 | 4005_B000 | 4005_BFFF |
| Flexible Timer Unit | FTU0 | | 60 | 4 | 4005_C000 | 4005_CFFF |
| | FTU1 | | 61 | 4 | 4005_D000 | 4005_DFFF |

Flagchip

**Table 4. APB bridge 0 (AFCB0) slot assignments (continued)**

| Peripheral Description | Peripheral Instance | SEG0 Slot | SEG1 Slot | Size (KB) | Start Address (hex) | End Address (hex) |
|---|---|---|---|---|---|---|
| | FTU2 | | 62 | 4 | 4005_E000 | 4005_EFFF |
| | FTU3 | | 63 | 4 | 4005_F000 | 4005_FFFF |
| Reserved | Reserved | | 64 | 4 | 4006_0000 | 4006_0FFF |
| Reserved | Reserved | | 65 | 4 | 4006_1000 | 4006_1FFF |
| FC Serial Peripheral Interface | FCSPI0 | | 66 | 4 | 4006_2000 | 4006_2FFF |
| | FCSPI1 | | 67 | 4 | 4006_3000 | 4006_3FFF |
| | FCSPI2 | | 68 | 4 | 4006_4000 | 4006_4FFF |
| Reserved | Reserved | | 69 | 4 | 4006_5000 | 4006_5FFF |
| FC Inter-Integrated Circuit | FCIIC0 | | 70 | 4 | 4006_6000 | 4006_6FFF |
| Reserved | Reserved | | 71 | 4 | 4006_7000 | 4006_7FFF |
| FC Universal Asynchronous Receiver/Transmitter | FCUART0 | | 72 | 4 | 4006_8000 | 4006_8FFF |
| | FCUART1 | | 73 | 4 | 4006_9000 | 4006_9FFF |
| | FCUART2 | | 74 | 4 | 4006_A000 | 4006_AFFF |
| | FCUART3 | | 75 | 4 | 4006_B000 | 4006_BFFF |
| Reserved | Reserved | | 76 | 4 | 4006_C000 | 4006_CFFF |
| Reserved | Reserved | | 77 | 4 | 4006_D000 | 4006_DFFF |
| Reserved | Reserved | | 78 | 4 | 4006_E000 | 4006_EFFF |
| Reserved | Reserved | | 79 | 4 | 4006_F000 | 4006_FFFF |
| Logic Unit | LU0 | | 80 | 4 | 4007_0000 | 4007_0FFF |
| Reserved | Reserved | | 81 | 4 | 4007_1000 | 4007_1FFF |
| System Control Module | SCM | | 82 | 4 | 4007_2000 | 4007_2FFF |
| Reserved | Reserved | | 83 | 4 | 4007_3000 | 4007_3FFF |
| Reserved | Reserved | | 84 | 4 | 4007_4000 | 4007_4FFF |
| Reserved | Reserved | | 85 | 4 | 4007_5000 | 4007_5FFF |
| Reserved | Reserved | | 86 | 4 | 4007_6000 | 4007_6FFF |
| Reserved | Reserved | | 87 | 4 | 4007_7000 | 4007_7FFF |
| Frequency Measurement | FREQM | | 88 | 4 | 4007_8000 | 4007_8FFF |
| CPU System Control | CSC0 | | 89 | 4 | 4007_9000 | 4007_9FFF |
| Reserved | Reserved | | 90 | 4 | 4007_A000 | 4007_AFFF |
| Reserved | Reserved | | 91 | 4 | 4007_B000 | 4007_BFFF |
| Reserved | Reserved | | 92 | 4 | 4007_C000 | 4007_CFFF |
| Reserved | Reserved | | 93 | 4 | 4007_D000 | 4007_DFFF |
| Reserved | Reserved | | 94 | 4 | 4007_E000 | 4007_EFFF |
| Safety Test and Control Unit | STCU | | 95 | 4 | 4007_F000 | 4007_FFFF |
| FLEX Controller Area Network | FLEXCAN0 | | 96 | 4 | 4008_0000 | 4008_0FFF |
| | | | 97 | 4 | 4008_1000 | 4008_1FFF |
| | | | 98 | 4 | 4008_2000 | 4008_2FFF |

**Table 4. APB bridge 0 (AFCB0) slot assignments (continued)**

| Peripheral Description | Peripheral Instance | SEG0 Slot | SEG1 Slot | Size (KB) | Start Address (hex) | End Address (hex) |
|---|---|---|---|---|---|---|
| | | | 99 | 4 | 4008_3000 | 4008_3FFF |
| FLEX Controller Area Network | FLEXCAN1 | | 100 | 4 | 4008_4000 | 4008_4FFF |
| | | | 101 | 4 | 4008_5000 | 4008_5FFF |
| | | | 102 | 4 | 4008_6000 | 4008_6FFF |
| | | | 103 | 4 | 4008_7000 | 4008_7FFF |
| Reserved | Reserved | | 104 | 4 | 4008_8000 | 4008_8FFF |
| | | | 105 | 4 | 4008_9000 | 4008_9FFF |
| | | | 106 | 4 | 4008_A000 | 4008_AFFF |
| | | | 107 | 4 | 4008_B000 | 4008_BFFF |
| Reserved | Reserved | | 108 | 4 | 4008_C000 | 4008_CFFF |
| | | | 109 | 4 | 4008_D000 | 4008_DFFF |
| | | | 110 | 4 | 4008_E000 | 4008_EFFF |
| | | | 111 | 4 | 4008_F000 | 4008_FFFF |
| Reserved | Reserved | | 112 | 4 | 4009_0000 | 4009_0FFF |
| | | | 113 | 4 | 4009_1000 | 4009_1FFF |
| | | | 114 | 4 | 4009_2000 | 4009_2FFF |
| | | | 115 | 4 | 4009_3000 | 4009_3FFF |
| Reserved | Reserved | | 116 | 4 | 4009_4000 | 4009_4FFF |
| | | | 117 | 4 | 4009_5000 | 4009_5FFF |
| | | | 118 | 4 | 4009_6000 | 4009_6FFF |
| | | | 119 | 4 | 4009_7000 | 4009_7FFF |
| Reserved | Reserved | | 120 | 4 | 4009_8000 | 4009_8FFF |
| | | | 121 | 4 | 4009_9000 | 4009_9FFF |
| | | | 122 | 4 | 4009_A000 | 4009_AFFF |
| | | | 123 | 4 | 4009_B000 | 4009_BFFF |
| Reserved | Reserved | | 124 | 4 | 4009_C000 | 4009_CFFF |
| | | | 125 | 4 | 4009_D000 | 4009_DFFF |
| | | | 126 | 4 | 4009_E000 | 4009_EFFF |
| | | | 127 | 4 | 4009_F000 | 4009_FFFF |

### 4.2.2 AFCB1 Slot Assignments

**Table 5. APB bridge 1 (AFCB1) slot assignments**

| Peripheral Description | Peripheral Instance | SEG0 Slot | SEG1 Slot | Size (KB) | Start Address (hex) | End Address (hex) | Note |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | 0 | | 4 | 4040_0000 | 4040_0FFF | |
| Reserved | Reserved | 1 | | 4 | 4040_1000 | 4040_1FFF | |
| Reserved | Reserved | 2 | | 4 | 4040_2000 | 4040_2FFF | |

**Table 5. APB bridge 1 (AFCB1) slot assignments (continued)**

| Peripheral Description | Peripheral Instance | SEG0 Slot | SEG1 Slot | Size (KB) | Start Address (hex) | End Address (hex) | Note |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | 3 | | 4 | 4040_3000 | 4040_3FFF | |
| Reserved | Reserved | 4 | | 4 | 4040_4000 | 4040_4FFF | |
| Reserved | Reserved | 5 | | 4 | 4040_5000 | 4040_5FFF | |
| Reserved | Reserved | 6 | | 4 | 4040_6000 | 4040_6FFF | |
| Reserved | Reserved | 7 | | 4 | 4040_7000 | 4040_7FFF | |
| Reserved | Reserved | 8 | | 4 | 4040_8000 | 4040_8FFF | |
| Reserved | Reserved | 9 | | 4 | 4040_9000 | 4040_9FFF | |
| Reserved | Reserved | 10 | | 4 | 4040_A000 | 4040_AFFF | |
| Reserved | Reserved | 11 | | 4 | 4040_B000 | 4040_BFFF | |
| Reserved | Reserved | 12 | | 4 | 4040_C000 | 4040_CFFF | |
| Matrix Access Monitor | MAM0 | 13 | | 4 | 4040_D000 | 4040_DFFF | |
| Reserved | Reserved | 14 | | 4 | 4040_E000 | 4040_EFFF | |
| Reserved | Reserved | 15 | | 4 | 4040_F000 | 4040_FFFF | |
| Reserved | Reserved | 16 | | 4 | 4041_0000 | 4041_0FFF | |
| Reserved | Reserved | 17 | | 4 | 4041_1000 | 4041_1FFF | |
| Reserved | Reserved | 18 | | 4 | 4041_2000 | 4041_2FFF | |
| Reserved | Reserved | 19 | | 4 | 4041_3000 | 4041_3FFF | |
| Reserved | Reserved | 20 | | 4 | 4041_4000 | 4041_4FFF | |
| Reserved | Reserved | 21 | | 4 | 4041_5000 | 4041_5FFF | |
| Reserved | Reserved | 22 | | 4 | 4041_6000 | 4041_6FFF | |
| Reserved | Reserved | 23 | | 4 | 4041_7000 | 4041_7FFF | |
| Reserved | Reserved | 24 | | 4 | 4041_8000 | 4041_8FFF | |
| Reserved | Reserved | 25 | | 4 | 4041_9000 | 4041_9FFF | |
| Reserved | Reserved | 26 | | 4 | 4041_A000 | 4041_AFFF | |
| Reserved | Reserved | 27 | | 4 | 4041_B000 | 4041_BFFF | |
| Reserved | Reserved | 28 | | 4 | 4041_C000 | 4041_CFFF | |
| Reserved | Reserved | 29 | | 4 | 4041_D000 | 4041_DFFF | |
| Reserved | Reserved | 30 | | 4 | 4041_E000 | 4041_EFFF | |
| Reserved | Reserved | 31 | | 4 | 4041_F000 | 4041_FFFF | |
| Reserved | Reserved | | 0 | 4 | 4042_0000 | 4042_0FFF | |
| Reserved | Reserved | | 1 | 4 | 4042_1000 | 4042_1FFF | |
| Reserved | Reserved | | 2 | 4 | 4042_2000 | 4042_2FFF | |
| Reserved | Reserved | | 3 | 4 | 4042_3000 | 4042_3FFF | |
| Reserved | Reserved | | 4 | 4 | 4042_4000 | 4042_4FFF | |
| Reserved | Reserved | | 5 | 4 | 4042_5000 | 4042_5FFF | |
| Reserved | Reserved | | 6 | 4 | 4042_6000 | 4042_6FFF | |
| Reserved | Reserved | | 7 | 4 | 4042_7000 | 4042_7FFF | |
| Reserved | Reserved | | 8 | 4 | 4042_8000 | 4042_8FFF | |

Flagchip

**Table 5. APB bridge 1 (AFCB1) slot assignments (continued)**

| Peripheral Description | Peripheral Instance | SEG0 Slot | SEG1 Slot | Size (KB) | Start Address (hex) | End Address (hex) | Note |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | | 9 | 4 | 4042_9000 | 4042_9FFF | |
| Reserved | Reserved | | 10 | 4 | 4042_A000 | 4042_AFFF | |
| Reserved | Reserved | | 11 | 4 | 4042_B000 | 4042_BFFF | |
| Reserved | Reserved | | 12 | 4 | 4042_C000 | 4042_CFFF | |
| Reserved | Reserved | | 13 | 4 | 4042_D000 | 4042_DFFF | |
| Reserved | Reserved | | 14 | 4 | 4042_E000 | 4042_EFFF | |
| Reserved | Reserved | | 15 | 4 | 4042_F000 | 4042_FFFF | |
| Reserved | Reserved | | 16 | 4 | 4043_0000 | 4043_0FFF | |
| Reserved | Reserved | | 17 | 4 | 4043_1000 | 4043_1FFF | |
| Reserved | Reserved | | 18 | 4 | 4043_2000 | 4043_2FFF | |
| Software Watchdog | WDOG1 | | 19 | 4 | 4043_3000 | 4043_3FFF | |
| Reserved | Reserved | | 20 | 4 | 4043_4000 | 4043_4FFF | |
| Reserved | Reserved | | 21 | 4 | 4043_5000 | 4043_5FFF | |
| Reserved | Reserved | | 22 | 4 | 4043_6000 | 4043_6FFF | |
| Reserved | Reserved | | 23 | 4 | 4043_7000 | 4043_7FFF | |
| Reserved | Reserved | | 24 | 4 | 4043_8000 | 4043_8FFF | |
| Reserved | Reserved | | 25 | 4 | 4043_9000 | 4043_9FFF | |
| Reserved | Reserved | | 26 | 4 | 4043_A000 | 4043_AFFF | |
| TRGSEL | TRGSEL4 | | 27 | 4 | 4043_B000 | 4043_BFFF | |
| | TRGSEL5 | | 28 | 4 | 4043_C000 | 4043_CFFF | |
| Reserved | Reserved | | 29 | 4 | 4043_D000 | 4043_DFFF | |
| Reserved | Reserved | | 30 | 4 | 4043_E000 | 4043_EFFF | |
| FC Serial Peripheral Interface | FCSPI3 | | 31 | 4 | 4043_F000 | 4043_FFFF | |
| | FCSPI4 | | 32 | 4 | 4044_0000 | 4044_0FFF | |
| | FCSPI5 | | 33 | 4 | 4044_1000 | 4044_1FFF | |
| Reserved | Reserved | | 34 | 4 | 4044_2000 | 4044_2FFF | |
| Reserved | Reserved | | 35 | 4 | 4044_3000 | 4044_3FFF | |
| Reserved | Reserved | | 36 | 4 | 4044_4000 | 4044_4FFF | |
| TPU Host | TPUH0 | | 37 | 4 | 4044_5000 | 4044_5FFF | |
| TPU Engine | TPUE0 | | 38 | 4 | 4044_6000 | 4044_6FFF | Only for non-FlexCore mode |
| Reserved | Reserved | | 39 | 4 | 4044_7000 | 4044_7FFF | |
| Reserved | Reserved | | 40 | 4 | 4044_8000 | 4044_8FFF | |
| Reserved | Reserved | | 41 | 4 | 4044_9000 | 4044_9FFF | |
| Reserved | Reserved | | 42 | 4 | 4044_A000 | 4044_AFFF | |
| Reserved | Reserved | | 43 | 4 | 4044_B000 | 4044_BFFF | |
| Reserved | Reserved | | 44 | 4 | 4044_C000 | 4044_CFFF | |

**Table 5. APB bridge 1 (AFCB1) slot assignments (continued)**

| Peripheral Description | Peripheral Instance | SEG0 Slot | SEG1 Slot | Size (KB) | Start Address (hex) | End Address (hex) | Note |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | | 45 | 4 | 4044_D000 | 4044_DFFF | |
| Reserved | Reserved | | 46 | 4 | 4044_E000 | 4044_EFFF | |
| Reserved | Reserved | | 47 | 4 | 4044_F000 | 4044_FFFF | |
| Reserved | Reserved | | 48 | 4 | 4045_0000 | 4045_0FFF | |
| Reserved | Reserved | | 49 | 4 | 4045_1000 | 4045_1FFF | |
| Reserved | Reserved | | 50 | 4 | 4045_2000 | 4045_2FFF | |
| Reserved | Reserved | | 51 | 4 | 4045_3000 | 4045_3FFF | |
| Reserved | Reserved | | 52 | 4 | 4045_4000 | 4045_4FFF | |
| Reserved | Reserved | | 53 | 4 | 4045_5000 | 4045_5FFF | |
| Reserved | Reserved | | 54 | 4 | 4045_6000 | 4045_6FFF | |
| Reserved | Reserved | | 55 | 4 | 4045_7000 | 4045_7FFF | |
| Reserved | Reserved | | 56 | 4 | 4045_8000 | 4045_8FFF | |
| Reserved | Reserved | | 57 | 4 | 4045_9000 | 4045_9FFF | |
| Reserved | Reserved | | 58 | 4 | 4045_A000 | 4045_AFFF | |
| Reserved | Reserved | | 59 | 4 | 4045_B000 | 4045_BFFF | |
| Reserved | Reserved | | 60 | 4 | 4045_C000 | 4045_CFFF | |
| Reserved | Reserved | | 61 | 4 | 4045_D000 | 4045_DFFF | |
| Reserved | Reserved | | 62 | 4 | 4045_E000 | 4045_EFFF | |
| Flexible Timer Unit | FTU4 | | 63 | 4 | 4045_F000 | 4045_FFFF | |
| | FTU5 | | 64 | 4 | 4046_0000 | 4046_0FFF | |
| | FTU6 | | 65 | 4 | 4046_1000 | 4046_1FFF | |
| | FTU7 | | 66 | 4 | 4046_2000 | 4046_2FFF | |
| Reserved | Reserved | | 67 | 4 | 4046_3000 | 4046_3FFF | |
| Reserved | Reserved | | 68 | 4 | 4046_4000 | 4046_4FFF | |
| Reserved | Reserved | | 69 | 4 | 4046_5000 | 4046_5FFF | |
| Reserved | Reserved | | 70 | 4 | 4046_6000 | 4046_6FFF | |
| FC Inter-Integrated Circuit | FCIIC1 | | 71 | 4 | 4046_7000 | 4046_7FFF | |
| FC Universal Asynchronous Receiver/Transmitter | FCUART4 | | 72 | 4 | 4046_8000 | 4046_8FFF | |
| | FCUART5 | | 73 | 4 | 4046_9000 | 4046_9FFF | |
| | FCUART6 | | 74 | 4 | 4046_A000 | 4046_AFFF | |
| | FCUART7 | | 75 | 4 | 4046_B000 | 4046_BFFF | |
| Reserved | Reserved | | 76 | 4 | 4046_C000 | 4046_CFFF | |
| Reserved | Reserved | | 77 | 4 | 4046_D000 | 4046_DFFF | |
| Reserved | Reserved | | 78 | 4 | 4046_E000 | 4046_EFFF | |
| Reserved | Reserved | | 79 | 4 | 4046_F000 | 4046_FFFF | |
| Reserved | Reserved | | 80 | 4 | 4047_0000 | 4047_0FFF | |
| Reserved | Reserved | | 81 | 4 | 4047_1000 | 4047_1FFF | |

**Table 5. APB bridge 1 (AFCB1) slot assignments (continued)**

| Peripheral Description | Peripheral Instance | SEG0 Slot | SEG1 Slot | Size (KB) | Start Address (hex) | End Address (hex) | Note |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | | 82 | 4 | 4047_2000 | 4047_2FFF | |
| Reserved | Reserved | | 83 | 4 | 4047_3000 | 4047_3FFF | |
| Micro Second Channel | MSC0 | | 84 | 4 | 4047_4000 | 4047_4FFF | |
| Reserved | Reserved | | 85 | 4 | 4047_5000 | 4047_5FFF | |
| Reserved | Reserved | | 86 | 4 | 4047_6000 | 4047_6FFF | |
| Reserved | Reserved | | 87 | 4 | 4047_7000 | 4047_7FFF | |
| Reserved | Reserved | | 88 | 4 | 4047_8000 | 4047_8FFF | |
| Reserved | Reserved | | 89 | 4 | 4047_9000 | 4047_9FFF | |
| Reserved | Reserved | | 90 | 4 | 4047_A000 | 4047_AFFF | |
| Reserved | Reserved | | 91 | 4 | 4047_B000 | 4047_BFFF | |
| Reserved | Reserved | | 92 | 4 | 4047_C000 | 4047_CFFF | |
| Reserved | Reserved | | 93 | 4 | 4047_D000 | 4047_DFFF | |
| Reserved | Reserved | | 94 | 4 | 4047_E000 | 4047_EFFF | |
| Reserved | Reserved | | 95 | 4 | 4047_F000 | 4047_FFFF | |
| FLEX Controller Area Network | FLEXCAN2 | | 96 | 4 | 4048_0000 | 4048_0FFF | |
| | | | 97 | 4 | 4048_1000 | 4048_1FFF | |
| | | | 98 | 4 | 4048_2000 | 4048_2FFF | |
| | | | 99 | 4 | 4048_3000 | 4048_3FFF | |
| FLEX Controller Area Network | FLEXCAN3 | | 100 | 4 | 4048_4000 | 4048_4FFF | |
| | | | 101 | 4 | 4048_5000 | 4048_5FFF | |
| | | | 102 | 4 | 4048_6000 | 4048_6FFF | |
| | | | 103 | 4 | 4048_7000 | 4048_7FFF | |
| Reserved | Reserved | | 104 | 4 | 4048_8000 | 4048_8FFF | |
| | | | 105 | 4 | 4048_9000 | 4048_9FFF | |
| | | | 106 | 4 | 4048_A000 | 4048_AFFF | |
| | | | 107 | 4 | 4048_B000 | 4048_BFFF | |
| Reserved | Reserved | | 108 | 4 | 4048_C000 | 4048_CFFF | |
| | | | 109 | 4 | 4048_D000 | 4048_DFFF | |
| | | | 110 | 4 | 4048_E000 | 4048_EFFF | |
| | | | 111 | 4 | 4048_F000 | 4048_FFFF | |
| Reserved | Reserved | | 112 | 4 | 4049_0000 | 4049_0FFF | |
| | | | 113 | 4 | 4049_1000 | 4049_1FFF | |
| | | | 114 | 4 | 4049_2000 | 4049_2FFF | |
| | | | 115 | 4 | 4049_3000 | 4049_3FFF | |
| Reserved | Reserved | | 116 | 4 | 4049_4000 | 4049_4FFF | |
| | | | 117 | 4 | 4049_5000 | 4049_5FFF | |
| | | | 118 | 4 | 4049_6000 | 4049_6FFF | |
| | | | 119 | 4 | 4049_7000 | 4049_7FFF | |

**Table 5. APB bridge 1 (AFCB1) slot assignments (continued)**

| Peripheral Description | Peripheral Instance | SEG0 Slot | SEG1 Slot | Size (KB) | Start Address (hex) | End Address (hex) | Note |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | | 120 | 4 | 4049_8000 | 4049_8FFF | |
| | | | 121 | 4 | 4049_9000 | 4049_9FFF | |
| | | | 122 | 4 | 4049_A000 | 4049_AFFF | |
| | | | 123 | 4 | 4049_B000 | 4049_BFFF | |
| HSM | HSM0 | | 124 | 4 | 4049_C000 | 4049_CFFF | |
| | | | 125 | 4 | 4049_D000 | 4049_DFFF | |
| | | | 126 | 4 | 4049_E000 | 4049_EFFF | |
| | | | 127 | 4 | 4049_F000 | 4049_FFFF | |

## 4.3 AHB Peripheral Assignments

The FC7240F2M has the below Advanced High-Performance Bus (AHB) peripheral memory map assignments.

**Table 6. AHB memory map assignments**

| AHB Peripheral Description | AHB Peripheral Instance | Size (KB) | Start Address (hex) | End Address (hex) |
|---|---|---|---|---|
| GPIO | GPIO | 64 | 4100_0000 | 4100_FFFF |
| Reserved | | 64 | 4101_0000 | 4101_FFFF |
| Reserved | | 64 | 4102_0000 | 4102_FFFF |
| Reserved | | 64 | 4103_0000 | 4103_FFFF |
| Reserved | | 64 | 4104_0000 | 4104_FFFF |
| Reserved | | 64 | 4105_0000 | 4105_FFFF |
| Reserved | | 64 | 4106_0000 | 4106_FFFF |
| Reserved | | 64 | 4107_0000 | 4107_FFFF |
| Reserved | | 64 | 4108_0000 | 4108_FFFF |

## 4.4 PPB Memory Map

The FC7240F2M has the below Private Peripheral Bus (PPB) IPs.

***Note:*** The Trace Port Interface Unit (TPIU)/FUNNEL etc. are only available on CM7_0.

**Table 7. FC7240F2M PPB memory map**

| PPB Peripheral Description | PPB Peripheral Instance | Size (KB) | Start Address (hex) | End Address (hex) | Comment |
|---|---|---|---|---|---|
| Instrumentation Trace Macrocell | ITM | 4 | E000_0000 | E000_0FFF | |
| Data Watchpoint and Trace | DWT | 4 | E000_1000 | E000_1FFF | |
| Flash Patch and Breakpoint | FPB | 4 | E000_2000 | E000_2FFF | |
| Reserved | Reserved | 44 | E000_3000 | E000_DFFF | |
| System Control Space | SCS | 4 | E000_E000 | E000_EFFF | |

**Table 7. FC7240F2M PPB memory map (continued)**

| PPB Peripheral Description | PPB Peripheral Instance | Size (KB) | Start Address (hex) | End Address (hex) | Comment |
|---|---|---|---|---|---|
| Reserved | Reserved | 196 | E000_F000 | E003_FFFF | |
| Trace Port Interface Unit | TPIU | 4 | E004_0000 | E004_0FFF | Only on Core0 |
| Embedded Trace Macro | ETM | 4 | E004_1000 | E004_1FFF | |
| Cross Trigger Interface | Reserved | 4 | E004_2000 | E004_2FFF | |
| Reserved | Reserved | 4 | E004_3000 | E004_3FFF | |
| Reserved | Reserved | 4 | E004_4000 | E004_4FFF | |
| Cross Trigger Interface | Reserved | 4 | E004_5000 | E004_5FFF | |
| Reserved | Reserved | 4 | E004_6000 | E004_6FFF | |
| Reserved | Reserved | 4 | E004_7000 | E004_7FFF | |
| Reserved | Reserved | 224 | E004_8000 | E007_FFFF | |
| CPU Private Module | CPM | 4 | E008_0000 | E008_0FFF | |
| AHB Overlay Module | AHB_OVERLAY | 4 | E008_1000 | E00F_CFFF | |
| Reserved | Reserved | 492 | E008_2000 | E00F_CFFF | |
| SYSTEM ROM Table | System Debug/Trace ROM Table | 4 | E00F_D000 | E00F_DFFF | Only on Core0 |
| CPU ROM Table | ARM CORE ROM Table | 4 | E00F_E000 | E00F_EFFF | |
| PPB ROM Table | PPB ROM Table | 4 | E00F_F000 | E00F_FFFF | |

## 4.5 Interrupt Vector Assignments

There is one CM7 core interrupt assignments for FC7240F2M. The below table lists the CM7_0 interrupt vector assignments.

**Table 8. CM7_0 Interrupt vector assignments**

| Vector Offset Address | Cortex -M7 Vector (0-142) | NVIC Interrupt ID | NVIC non-IPR Register | NVIC IPR Register | Name | Instance | Interrupt Type |
|---|---|---|---|---|---|---|---|
| **ARM Core System Handler Vector** | | | | | | | |
| 0x0000_0000 | 0 | - | - | - | Initial Stack Pointer | ARM_CORE | Initial Stack Pointer |
| 0x0000_0004 | 1 | - | - | - | Initial Program Counter | ARM_CORE | Initial Program Counter |
| 0x0000_0008 | 2 | - | - | - | Non-maskable Interrupt (NMI) | ARM_CORE | Non-maskable Interrupt (NMI) |
| 0x0000_000C | 3 | - | - | - | Hard Fault | ARM_CORE | Hard Fault |
| 0x0000_0010 | 4 | - | - | - | MemManage Fault | ARM_CORE | MemManage Fault |
| 0x0000_0014 | 5 | - | - | - | Bus Fault | ARM_CORE | Bus Fault |
| 0x0000_0018 | 6 | - | - | - | Usage Fault | ARM_CORE | Usage Fault |
| 0x0000_001C | 7 | - | - | - | Reserved | Reserved | Reserved |
| 0x0000_0020 | 8 | - | - | - | Reserved | Reserved | Reserved |

**Table 8. Interrupt vector assignments (continued)**

| Vector Offset Address | Cortex -M7 Vector (0-142) | NVIC Interrupt ID | NVIC non-IPR Register | NVIC IPR Register | Name | Instance | Interrupt Type |
|---|---|---|---|---|---|---|---|
| 0x0000_0024 | 9 | - | - | - | Reserved | Reserved | Reserved |
| 0x0000_0028 | 10 | - | - | - | Reserved | Reserved | Reserved |
| 0x0000_002C | 11 | - | - | - | Supervisor call (SVCall) | ARM_CORE | Supervisor call (SVCall) |
| 0x0000_0030 | 12 | - | - | - | Debug Monitor | ARM_CORE | Debug Monitor |
| 0x0000_0034 | 13 | - | - | - | Reserved | Reserved | Reserved |
| 0x0000_0038 | 14 | - | - | - | Pendable request for system service (PendableSrvReq) | ARM_CORE | Pendable request for system service (PendableSrvReq) |
| 0x0000_003C | 15 | - | - | - | System Tick Timer (SysTick) | ARM_CORE | System Tick Timer (SysTick) |
| **Non-Core Vectors** | | | | | | | |
| 0x0000_0040 | 16 | 0 | NVIC_xxxx0 [0] | NVIC_IPR0 [7:4] | DMA | DMA0 | DMA channel 0 transfer complete |
| 0x0000_0044 | 17 | 1 | NVIC_xxxx0 [1] | NVIC_IPR0 [15:12] | DMA | DMA0 | DMA channel 1 transfer complete |
| 0x0000_0048 | 18 | 2 | NVIC_xxxx0 [2] | NVIC_IPR0 [23:20] | DMA | DMA0 | DMA channel 2 transfer complete |
| 0x0000_004C | 19 | 3 | NVIC_xxxx0 [3] | NVIC_IPR0 [31:28] | DMA | DMA0 | DMA channel 3 transfer complete |
| 0x0000_0050 | 20 | 4 | NVIC_xxxx0 [4] | NVIC_IPR1 [7:4] | DMA | DMA0 | DMA channel 4 transfer complete |
| 0x0000_0054 | 21 | 5 | NVIC_xxxx0 [5] | NVIC_IPR1 [15:12] | DMA | DMA0 | DMA channel 5 transfer complete |
| 0x0000_0058 | 22 | 6 | NVIC_xxxx0 [6] | NVIC_IPR1 [23:20] | DMA | DMA0 | DMA channel 6 transfer complete |
| 0x0000_005C | 23 | 7 | NVIC_xxxx0 [7] | NVIC_IPR1 [31:28] | DMA | DMA0 | DMA channel 7 transfer complete |
| 0x0000_0060 | 24 | 8 | NVIC_xxxx0 [8] | NVIC_IPR2 [7:4] | DMA | DMA0 | DMA channel 8 transfer complete |
| 0x0000_0064 | 25 | 9 | NVIC_xxxx0 [9] | NVIC_IPR2 [15:12] | DMA | DMA0 | DMA channel 9 transfer complete |
| 0x0000_0068 | 26 | 10 | NVIC_xxxx0 [10] | NVIC_IPR2 [23:20] | DMA | DMA0 | DMA channel 10 transfer complete |
| 0x0000_006C | 27 | 11 | NVIC_xxxx0 [11] | NVIC_IPR2 [31:28] | DMA | DMA0 | DMA channel 11 transfer complete |

**Table 8. Interrupt vector assignments (continued)**

| Vector Offset Address | Cortex -M7 Vector (0-142) | NVIC Interrupt ID | NVIC non-IPR Register | NVIC IPR Register | Name | Instance | Interrupt Type |
|---|---|---|---|---|---|---|---|
| 0x0000_0070 | 28 | 12 | NVIC_xxxx0 [12] | NVIC_IPR3 [7:4] | DMA | DMA0 | DMA channel 12 transfer complete |
| 0x0000_0074 | 29 | 13 | NVIC_xxxx0 [13] | NVIC_IPR3 [15:12] | DMA | DMA0 | DMA channel 13 transfer complete |
| 0x0000_0078 | 30 | 14 | NVIC_xxxx0 [14] | NVIC_IPR3 [23:20] | DMA | DMA0 | DMA channel 14 transfer complete |
| 0x0000_007C | 31 | 15 | NVIC_xxxx0 [15] | NVIC_IPR3 [31:28] | DMA | DMA0 | DMA channel 15 transfer complete |
| 0x0000_0080 | 32 | 16 | NVIC_xxxx0 [16] | NVIC_IPR4 [7:4] | DMA | DMA0 | DMA error interrupt channels 0-31 |
| 0x0000_0084 | 33 | 17 | NVIC_xxxx0 [17] | NVIC_IPR4 [15:12] | CPM | CPM | FPU etc. interrupt |
| 0x0000_0088 | 34 | 18 | NVIC_xxxx0 [18] | NVIC_IPR4 [23:20] | FC | FC | Flash Controller Command complete, time out etc. interrupt |
| 0x0000_008C | 35 | 19 | NVIC_xxxx0 [19] | NVIC_IPR4 [31:28] | PMC | PMC | HVD/LVD etc. interrupt |
| 0x0000_0090 | 36 | 20 | NVIC_xxxx0 [20] | NVIC_IPR5 [7:4] | TMU | TMU | Temperature Monitor Unit interrupt |
| 0x0000_0094 | 37 | 21 | NVIC_xxxx0 [21] | NVIC_IPR5 [15:12] | WDOG | WDOG0 | Interrupt request out before wdg reset out |
| 0x0000_0098 | 38 | 22 | NVIC_xxxx0 [22] | NVIC_IPR5 [23:20] | WDOG | WDOG1 | Interrupt request out before wdg reset out |
| 0x0000_009C | 39 | 23 | NVIC_xxxx0 [23] | NVIC_IPR5 [31:28] | FCSMU | FCSMU | Fault Control and Safety Manage Unit |
| 0x0000_00A0 | 40 | 24 | NVIC_xxxx0 [24] | NVIC_IPR6 [7:4] | STCU | STCU | Safety Control Unit interrupt |
| 0x0000_00A4 | 41 | 25 | NVIC_xxxx0 [25] | NVIC_IPR6 [15:12] | ERM | ERM | ERM single- or double-bit error interrupt |
| 0x0000_00A8 | 42 | 26 | NVIC_xxxx0 [26] | NVIC_IPR6 [23:20] | MAM | MAM0 | Matrix Access Monitor interrupt |
| 0x0000_00AC | 43 | 27 | NVIC_xxxx0 [27] | NVIC_IPR6 [31:28] | RGM | RGM | RGM pre-reset Interrupt |
| 0x0000_00B0 | 44 | 28 | NVIC_xxxx0 [28] | NVIC_IPR7 [7:4] | INTM | INTM0 | INTM timeout interrupt |
| 0x0000_00B4 | 45 | 29 | NVIC_xxxx0 [29] | NVIC_IPR7 [15:12] | ISM | ISM0 | ISM0 interrupt |

**Table 8. Interrupt vector assignments (continued)**

| Vector Offset Address | Cortex -M7 Vector (0-142) | NVIC Interrupt ID | NVIC non-IPR Register | NVIC IPR Register | Name | Instance | Interrupt Type |
|---|---|---|---|---|---|---|---|
| 0x0000_00B8 | 46 | 30 | NVIC_xxxx0 [30] | NVIC_IPR7 [23:20] | MB | MB | Mail Box interrupt |
| 0x0000_00BC | 47 | 31 | NVIC_xxxx0 [3] | NVIC_IPR7 [31:28] | SCG | SCG | SCG bus interrupt request |
| 0x0000_00C0 | 48 | 32 | NVIC_xxxx1 [0] | NVIC_IPR8 [7:4] | CMU | CMU0 | CMU0 interrupt |
| 0x0000_00C4 | 49 | 33 | NVIC_xxxx1 [1] | NVIC_IPR8 [15:12] | CMU | CMU1 | CMU1 interrupt |
| 0x0000_00C8 | 50 | 34 | NVIC_xxxx1 [2] | NVIC_IPR8 [23:20] | CMU | CMU2 | CMU2 interrupt |
| 0x0000_00CC | 51 | 35 | NVIC_xxxx1 [3] | NVIC_IPR8 [31:28] | CMU | CMU3 | CMU3 interrupt |
| 0x0000_00D0 | 52 | 36 | NVIC_xxxx1 [4] | NVIC_IPR9 [7:4] | CMU | CMU4 | CMU4 interrupt |
| 0x0000_00D4 | 53 | 37 | NVIC_xxxx1 [5] | NVIC_IPR9 [15:12] | TSTAMP | TSTAMP0 | TimerStamp interrupt |
| 0x0000_00D8 | 54 | 38 | NVIC_xxxx1 [6] | NVIC_IPR9 [23:20] | TSTAMP | TSTAMP1 | TimerStamp interrupt |
| 0x0000_00DC | 55 | 39 | NVIC_xxxx1 [7] | NVIC_IPR9 [31:28] | CORDIC | CORDIC0 | CORDIC Accelerator interrupt |
| 0x0000_00E0 | 56 | 40 | NVIC_xxxx1 [8] | NVIC_IPR 10[7:4] | HSM | HSM0 | HSM error interrupt |
| 0x0000_00E4 | 57 | 41 | NVIC_xxxx1 [9] | NVIC_IPR 10[15:12] | FCPIT | FCPIT0 | FCPIT interrupt |
| 0x0000_00E8 | 58 | 42 | NVIC_xxxx1 [10] | NVIC_IPR 10[23:20] | RTC | RTC | RTC alarm or seconds interrupt |
| 0x0000_00EC | 59 | 43 | NVIC_xxxx1 [11] | NVIC_IPR 10[31:28] | AONTIMER | AONTIMER0 | AONTIMER interrupt request |
| 0x0000_00F0 | 60 | 44 | NVIC_xxxx1 [12] | NVIC_IPR 11[7:4] | Software | Software | Software interrupt |
| 0x0000_00F4 | 61 | 45 | NVIC_xxxx1 [13] | NVIC_IPR 11[15:12] | FREQM | FREQM | FREQM interrupt |
| 0x0000_00F8 | 62 | 46 | NVIC_xxxx1 [14] | NVIC_IPR 11[23:20] | ADC | ADC0 | ADC interrupt request. |
| 0x0000_00FC | 63 | 47 | NVIC_xxxx1 [15] | NVIC_IPR 11[31:28] | ADC | ADC1 | ADC interrupt request. |
| 0x0000_0100 | 64 | 48 | NVIC_xxxx1 [16] | NVIC_IPR 12[7:4] | PTIMER | PTIMER0 | PTIMER interrupt |
| 0x0000_0104 | 65 | 49 | NVIC_xxxx1 [17] | NVIC_IPR 12[15:12] | PTIMER | PTIMER1 | PTIMER interrupt |
| 0x0000_0108 | 66 | 50 | NVIC_xxxx1 [18] | NVIC_IPR 12[23:20] | FLEXCAN | FLEXCAN0 | FLEXCAN interrupt |

**Table 8. Interrupt vector assignments (continued)**

| Vector Offset Address | Cortex -M7 Vector (0-142) | NVIC Interrupt ID | NVIC non-IPR Register | NVIC IPR Register | Name | Instance | Interrupt Type |
|---|---|---|---|---|---|---|---|
| 0x0000_010C | 67 | 51 | NVIC_xxxx1 [19] | NVIC_IPR 12[31:28] | FLEXCAN | FLEXCAN1 | FLEXCAN interrupt |
| 0x0000_0110 | 68 | 52 | NVIC_xxxx1 [20] | NVIC_IPR 13[7:4] | FLEXCAN | FLEXCAN2 | FLEXCAN interrupt |
| 0x0000_0114 | 69 | 53 | NVIC_xxxx1 [21] | NVIC_IPR 13[15:12] | FLEXCAN | FLEXCAN3 | FLEXCAN interrupt |
| 0x0000_0118 | 70 | 54 | NVIC_xxxx1 [22] | NVIC_IPR 13[23:20] | FCIIC | FCIIC0 | FCIIC Interrupt |
| 0x0000_011C | 71 | 55 | NVIC_xxxx1 [23] | NVIC_IPR 13[31:28] | FCIIC | FCIIC1 | FCIIC Interrupt |
| 0x0000_0120 | 72 | 56 | NVIC_xxxx1 [24] | NVIC_IPR 14[7:4] | FCSPI | FCSPI0 | FCSPI Interrupt |
| 0x0000_0124 | 73 | 57 | NVIC_xxxx1 [25] | NVIC_IPR 14[15:12] | FCSPI | FCSPI1 | FCSPI Interrupt |
| 0x0000_0128 | 74 | 58 | NVIC_xxxx1 [26] | NVIC_IPR 14[23:20] | FCSPI | FCSPI2 | FCSPI Interrupt |
| 0x0000_012C | 75 | 59 | NVIC_xxxx1 [27] | NVIC_IPR 14[31:28] | FCSPI | FCSPI3 | FCSPI Interrupt |
| 0x0000_0130 | 76 | 60 | NVIC_xxxx1 [28] | NVIC_IPR 15[7:4] | FCSPI | FCSPI4 | FCSPI Interrupt |
| 0x0000_0134 | 77 | 61 | NVIC_xxxx1 [29] | NVIC_IPR 15[15:12] | FCSPI | FCSPI5 | FCSPI Interrupt |
| 0x0000_0138 | 78 | 62 | NVIC_xxxx1 [30] | NVIC_IPR 15[23:20] | FCUART | FCUART0 | FCUART Interrupt |
| 0x0000_013C | 79 | 63 | NVIC_xxxx1 [31] | NVIC_IPR 15[31:28] | FCUART | FCUART1 | FCUART Interrupt |
| 0x0000_0140 | 80 | 64 | NVIC_xxxx2 [0] | NVIC_IPR 16[7:4] | FCUART | FCUART2 | FCUART Interrupt |
| 0x0000_0144 | 81 | 65 | NVIC_xxxx2 [1] | NVIC_IPR 16[15:12] | FCUART | FCUART3 | FCUART Interrupt |
| 0x0000_0148 | 82 | 66 | NVIC_xxxx2 [2] | NVIC_IPR 16[23:20] | FCUART | FCUART4 | FCUART Interrupt |
| 0x0000_014C | 83 | 67 | NVIC_xxxx2 [3] | NVIC_IPR 16[31:28] | FCUART | FCUART5 | FCUART Interrupt |
| 0x0000_0150 | 84 | 68 | NVIC_xxxx2 [4] | NVIC_IPR 17[7:4] | FCUART | FCUART6 | FCUART Interrupt |
| 0x0000_0154 | 85 | 69 | NVIC_xxxx2 [5] | NVIC_IPR 17[15:12] | FCUART | FCUART7 | FCUART Interrupt |
| 0x0000_0158 | 86 | 70 | NVIC_xxxx2 [6] | NVIC_IPR 17[23:20] | FTU | FTU0 | FTU all source interrupt |
| 0x0000_015C | 87 | 71 | NVIC_xxxx2 [7] | NVIC_IPR 17[31:28] | FTU | FTU1 | FTU all source interrupt |

**Table 8. Interrupt vector assignments (continued)**

| Vector Offset Address | Cortex -M7 Vector (0-142) | NVIC Interrupt ID | NVIC non-IPR Register | NVIC IPR Register | Name | Instance | Interrupt Type |
|---|---|---|---|---|---|---|---|
| 0x0000_0160 | 88 | 72 | NVIC_xxxx2 [8] | NVIC_IPR 18[7:4] | FTU | FTU2 | FTU all source interrupt |
| 0x0000_0164 | 89 | 73 | NVIC_xxxx2 [9] | NVIC_IPR 18[15:12] | FTU | FTU3 | FTU all source interrupt |
| 0x0000_0168 | 90 | 74 | NVIC_xxxx2 [10] | NVIC_IPR 18[23:20] | FTU | FTU4 | FTU all source interrupt |
| 0x0000_016C | 91 | 75 | NVIC_xxxx2 [11] | NVIC_IPR 18[31:28] | FTU | FTU5 | FTU all source interrupt |
| 0x0000_0170 | 92 | 76 | NVIC_xxxx2 [12] | NVIC_IPR 19[7:4] | FTU | FTU6 | FTU all source interrupt |
| 0x0000_0174 | 93 | 77 | NVIC_xxxx2 [13] | NVIC_IPR 19[15:12] | FTU | FTU7 | FTU all source interrupt |
| 0x0000_0178 | 94 | 78 | NVIC_xxxx2 [14] | NVIC_IPR 19[23:20] | CMP | CMP0 | CMP interrupt request |
| 0x0000_017C | 95 | 79 | NVIC_xxxx2 [15] | NVIC_IPR 19[31:28] | CMP | CMP1 | CMP interrupt request |
| 0x0000_0180 | 96 | 80 | NVIC_xxxx2 [16] | NVIC_IPR 20[7:4] | Port Control | PORTA | PORTA Interrupt |
| 0x0000_0184 | 97 | 81 | NVIC_xxxx2 [17] | NVIC_IPR 20[15:12] | Port Control | PORTB | PORTB Interrupt |
| 0x0000_0188 | 98 | 82 | NVIC_xxxx2 [18] | NVIC_IPR 20[23:20] | Port Control | PORTC | PORTC Interrupt |
| 0x0000_018C | 99 | 83 | NVIC_xxxx2 [19] | NVIC_IPR 20[31:28] | Port Control | PORTD | PORTD Interrupt |
| 0x0000_0190 | 100 | 84 | NVIC_xxxx2 [20] | NVIC_IPR 21[7:4] | Port Control | PORTE | PORTE Interrupt |
| 0x0000_0194 | 101 | 85 | NVIC_xxxx2 [21] | NVIC_IPR 21[15:12] | MSC | MSC0 | MSC interrupt |
| 0x0000_0198 | 102 | 86 | NVIC_xxxx2 [22] | NVIC_IPR 21[23:20] | SENT | SENT0 | SENT all interrupt (fast or slow) |
| 0x0000_019C | 103 | 87 | NVIC_xxxx2 [23] | NVIC_IPR 21[31:28] | TPU | TPU0 | TPU0 CH0-7 interrupt |
| 0x0000_01A0 | 104 | 88 | NVIC_xxxx2 [24] | NVIC_IPR 22[7:4] | TPU | TPU0 | TPU0 CH8-15 interrupt |
| 0x0000_01A4 | 105 | 89 | NVIC_xxxx2 [25] | NVIC_IPR 22[15:12] | TPU | TPU0 | TPU0 CH16-23 interrupt |
| 0x0000_01A8 | 106 | 90 | NVIC_xxxx2 [26] | NVIC_IPR 22[23:20] | TPU | TPU0 | TPU0 CH24-31 interrupt |

Flagchip

**Table 8. Interrupt vector assignments (continued)**

| Vector Offset Address | Cortex -M7 Vector (0-142) | NVIC Interrupt ID | NVIC non-IPR Register | NVIC IPR Register | Name | Instance | Interrupt Type |
|---|---|---|---|---|---|---|---|
| 0x0000_01AC | 107 | 91 | NVIC_xxxx2 [27] | NVIC_IPR 22[31:28] | HSM | HSM0 | HSM crypto interrupt<br>***Note:*** Only for non-FlexCore mode |

### 4.5.1 WIC Wake-up Sources

**Table 9. WIC stop wake-up sources**

| Wake-up Sources | Description |
|---|---|
| RESET pin, WDOG, JTAG | Reset trigger sources, including the RESET pin, WDOG and JTAG |
| Pin interrupts | Pin interrupts in Port Control Module - Any enabled pin interrupt can wake up the system (for standby mode, only PD0 domain pin can be used to wake up the system) |
| ADCx | Async Interrupt in STOP mode |
| CMPx | Async Interrupt in low power mode |
| FCUARTx | Async Interrupt in low power mode |
| FCSPIx | Async Interrupt in low power mode |
| AONTIMERx | Async Interrupt in low power mode |
| FCPITx | Async Interrupt in low power mode |
| FLEXCANx | Async Interrupt in STOP mode |

***Note:*** For standby wakeup, only IPs in standby domain can wakeup system.

## 4.6 DMAMUX Assignments

The FC7240F2M contains one Direct Memory Access Multiplexer (DMAMUX). The below table lists the DMA channel assignments.

**Table 10. DMAMUX assignments**

| Slot_Num | Module | DMA_Request | Instance |
|---|---|---|---|
| 0 | Reserved | Reserved | Reserved |
| 1 | FCIIC | FCIIC DMA RX Request | FCIIC0 |
| 2 | FCIIC | FCIIC DMA TX Request | FCIIC0 |
| 3 | FCIIC | FCIIC DMA RX Request | FCIIC1 |
| 4 | FCIIC | FCIIC DMA TX Request | FCIIC1 |
| 5 | FCSPI | FCSPI DMA RX Request | FCSPI0 |
| 6 | FCSPI | FCSPI DMA TX Request | FCSPI0 |
| 7 | FCSPI | FCSPI DMA RX Request | FCSPI1 |
| 8 | FCSPI | FCSPI DMA TX Request | FCSPI1 |
| 9 | FCSPI | FCSPI DMA RX Request | FCSPI2 |

**Table 10. DMAMUX assignments (continued)**

| Slot_Num | Module | DMA_Request | Instance |
|---|---|---|---|
| 10 | FCSPI | FCSPI DMA TX Request | FCSPI2 |
| 11 | FCSPI | FCSPI DMA RX Request | FCSPI3 |
| 12 | FCSPI | FCSPI DMA TX Request | FCSPI3 |
| 13 | FCSPI | FCSPI DMA RX Request | FCSPI4 |
| 14 | FCSPI | FCSPI DMA TX Request | FCSPI4 |
| 15 | FCSPI | FCSPI DMA RX Request | FCSPI5 |
| 16 | FCSPI | FCSPI DMA TX Request | FCSPI5 |
| 17 | Reserved | Reserved | Reserved |
| 18 | Reserved | Reserved | Reserved |
| 19 | Reserved | Reserved | Reserved |
| 20 | Reserved | Reserved | Reserved |
| 21 | Reserved | Reserved | Reserved |
| 22 | Reserved | Reserved | Reserved |
| 23 | FLEXCAN | FLEXCAN DMA Request | FLEXCAN0 |
| 24 | FLEXCAN | FLEXCAN DMA Request | FLEXCAN1 |
| 25 | FLEXCAN | FLEXCAN DMA Request | FLEXCAN2 |
| 26 | FLEXCAN | FLEXCAN DMA Request | FLEXCAN3 |
| 27 | Reserved | Reserved | Reserved |
| 28 | Reserved | Reserved | Reserved |
| 29 | Reserved | Reserved | Reserved |
| 30 | Reserved | Reserved | Reserved |
| 31 | Reserved | Reserved | Reserved |
| 32 | Reserved | Reserved | Reserved |
| 33 | ADC | ADC DMA request | ADC0 |
| 34 | ADC | ADC DMA request | ADC1 |
| 35 | Reserved | Reserved | Reserved |
| 36 | Reserved | Reserved | Reserved |
| 37 | Reserved | Reserved | Reserved |
| 38 | Reserved | Reserved | Reserved |
| 39 | Reserved | Reserved | Reserved |
| 40 | Reserved | Reserved | Reserved |
| 41 | FCUART | FCUART Receive DMA Request | FCUART0 |
| 42 | FCUART | FCUART Transmit DMA Request | FCUART0 |
| 43 | FCUART | FCUART Receive DMA Request | FCUART1 |
| 44 | FCUART | FCUART Transmit DMA Request | FCUART1 |
| 45 | FCUART | FCUART Receive DMA Request | FCUART2 |
| 46 | FCUART | FCUART Transmit DMA Request | FCUART2 |
| 47 | FCUART | FCUART Receive DMA Request | FCUART3 |
| 48 | FCUART | FCUART Transmit DMA Request | FCUART3 |

**Table 10. DMAMUX assignments (continued)**

| Slot_Num | Module | DMA_Request | Instance |
|---|---|---|---|
| 49 | FCUART | FCUART Receive DMA Request | FCUART4 |
| 50 | FCUART | FCUART Transmit DMA Request | FCUART4 |
| 51 | FCUART | FCUART Receive DMA Request | FCUART5 |
| 52 | FCUART | FCUART Transmit DMA Request | FCUART5 |
| 53 | FCUART | FCUART Receive DMA Request | FCUART6 |
| 54 | FCUART | FCUART Transmit DMA Request | FCUART6 |
| 55 | FCUART | FCUART Receive DMA Request | FCUART7 |
| 56 | FCUART | FCUART Transmit DMA Request | FCUART7 |
| 57 | Reserved | Reserved | Reserved |
| 58 | Reserved | Reserved | Reserved |
| 59 | TPU | TPU channel 0 | TPU0 |
| 60 | TPU | TPU channel 1 | TPU0 |
| 61 | TPU | TPU channel 2 | TPU0 |
| 62 | TPU | TPU channel 3 | TPU0 |
| 63 | TPU | TPU channel 4 | TPU0 |
| 64 | TPU | TPU channel 5 | TPU0 |
| 65 | TPU | TPU channel 6 | TPU0 |
| 66 | TPU | TPU channel 7 | TPU0 |
| 67 | TPU | TPU channel 8 | TPU0 |
| 68 | TPU | TPU channel 9 | TPU0 |
| 69 | TPU | TPU channel 10 | TPU0 |
| 70 | TPU | TPU channel 11 | TPU0 |
| 71 | TPU | TPU channel 12 | TPU0 |
| 72 | TPU | TPU channel 13 | TPU0 |
| 73 | TPU | TPU channel 14 | TPU0 |
| 74 | TPU | TPU channel 15 | TPU0 |
| 75 | TPU | TPU channel 16-23 | TPU0 |
| 76 | TPU | TPU channel 24-31 | TPU0 |
| 77 | Port Control | PORTA DMA request | PORTA |
| 78 | Port Control | PORTB DMA request | PORTB |
| 79 | Port Control | PORTC DMA request | PORTC |
| 80 | Port Control | PORTD DMA request | PORTD |
| 81 | Port Control | PORTE DMA request | PORTE |
| 82 | Reserved | Reserved | Reserved |
| 83 | Reserved | Reserved | Reserved |
| 84 | Reserved | Reserved | Reserved |
| 85 | Reserved | Reserved | Reserved |
| 86 | AONTIMER | AONTIMER DMA request | AONTIMER0 |
| 87 | CMP | CMP DMA request | CMP0 |

**Table 10. DMAMUX assignments (continued)**

| Slot_Num | Module | DMA_Request | Instance |
|---|---|---|---|
| 88 | CMP | CMP DMA request | CMP1 |
| 89 | Reserved | Reserved | Reserved |
| 90 | PTIMER | PTIMER DMA request | PTIMER0 |
| 91 | PTIMER | PTIMER DMA request | PTIMER1 |
| 92 | Reserved | Reserved | Reserved |
| 93 | Reserved | Reserved | Reserved |
| 94 | Reserved | Reserved | Reserved |
| 95 | Reserved | Reserved | Reserved |
| 96 | Reserved | Reserved | Reserved |
| 97 | Reserved | Reserved | Reserved |
| 98 | FTU | FTU all channels OR DMA request | FTU0 |
| 99 | FTU | FTU all channels OR DMA request | FTU1 |
| 100 | FTU | FTU all channels OR DMA request | FTU2 |
| 101 | FTU | FTU all channels OR DMA request | FTU3 |
| 102 | FTU | FTU all channels OR DMA request | FTU4 |
| 103 | FTU | FTU all channels OR DMA request | FTU5 |
| 104 | FTU | FTU all channels OR DMA request | FTU6 |
| 105 | FTU | FTU all channels OR DMA request | FTU7 |
| 106 | Reserved | Reserved | Reserved |
| 107 | Reserved | Reserved | Reserved |
| 108 | Reserved | Reserved | Reserved |
| 109 | Reserved | Reserved | Reserved |
| 110 | SENT | SENT Channel 0 Fast DMA request | SENT0 |
| 111 | SENT | SENT Channel 1 Fast DMA request | SENT0 |
| 112 | SENT | SENT Channel 2 Fast DMA request | SENT0 |
| 113 | SENT | SENT Channel 3 Fast DMA request | SENT0 |
| 114 | Reserved | Reserved | Reserved |
| 115 | Reserved | Reserved | Reserved |
| 116 | Reserved | Reserved | Reserved |
| 117 | Reserved | Reserved | Reserved |
| 118 | SENT | SENT Channel 0 Slow DMA request | SENT0 |
| 119 | SENT | SENT Channel 1 Slow DMA request | SENT0 |
| 120 | SENT | SENT Channel 2 Slow DMA request | SENT0 |
| 121 | SENT | SENT Channel 3 Slow DMA request | SENT0 |
| 122 | Reserved | Reserved | Reserved |
| 123 | Reserved | Reserved | Reserved |
| 124 | Reserved | Reserved | Reserved |
| 125 | Reserved | Reserved | Reserved |
| 126 | ALWAYS ENABLED | | |

**Table 10. DMAMUX assignments (continued)**

| Slot_Num | Module | DMA_Request | Instance |
|---|---|---|---|
| 127 | ALWAYS ENABLED | | |

Flagchip

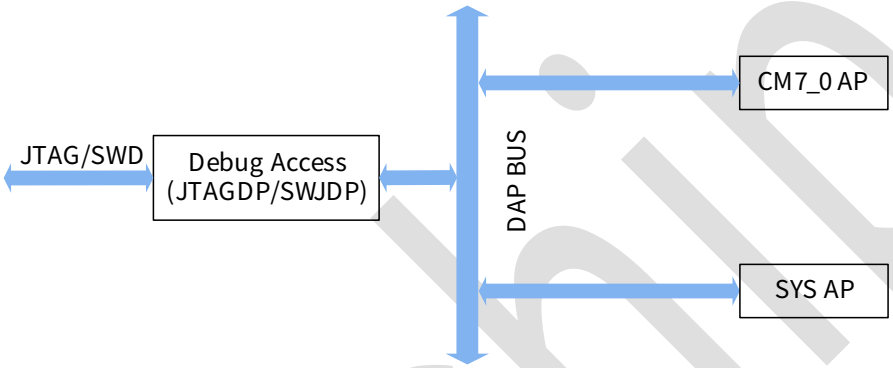
# Chapter 5 Debug and Trace

The FC7240F2M supports both Serial Wire Debug (SWD) and JTAG interfaces for the Cortex-M7 access. It has also an extra SYS-AP to support the system-level debug feature.

The FC7240F2M JTAG ID is {4'b0000, 6'b000001, 10'b0000000111, 11'b1111_1100_000, 1'b1}

The below diagram shows the FC7240F2M debug access port (AP) connection.

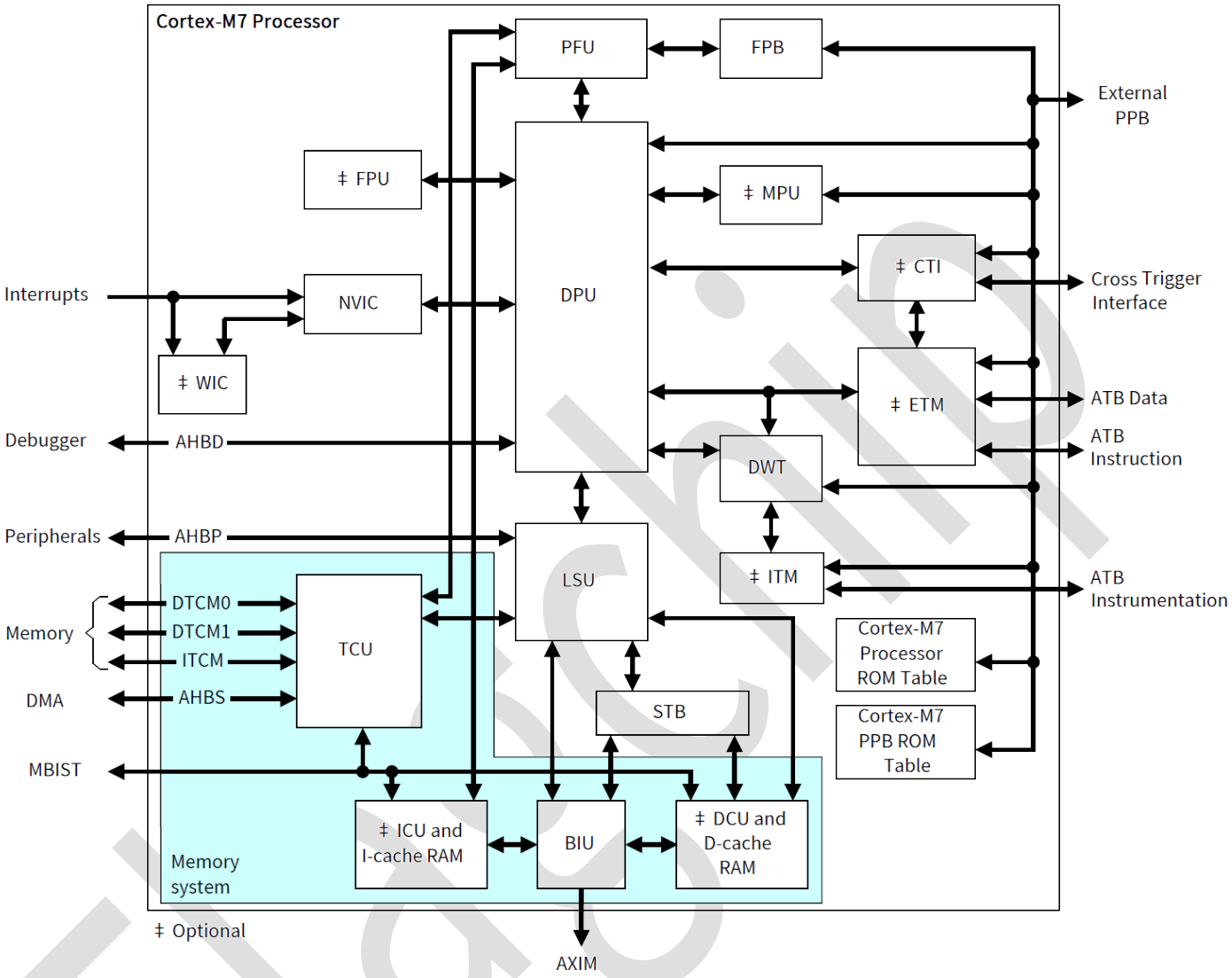
**Figure 3. Debug AP system**




Flagchip

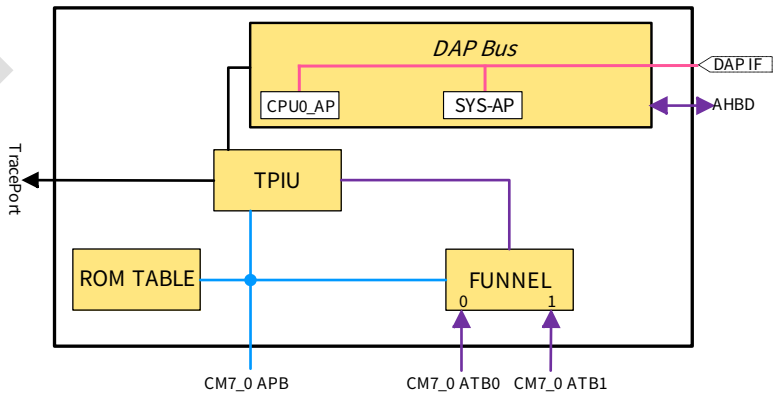
The below diagram shows the FC7240F2M Cortex-M7 debug structure. It has dedicated debug components including ETM/TPIU/DWT/FPB/ITM.

**Figure 4. Cortex-M7 debug structure**



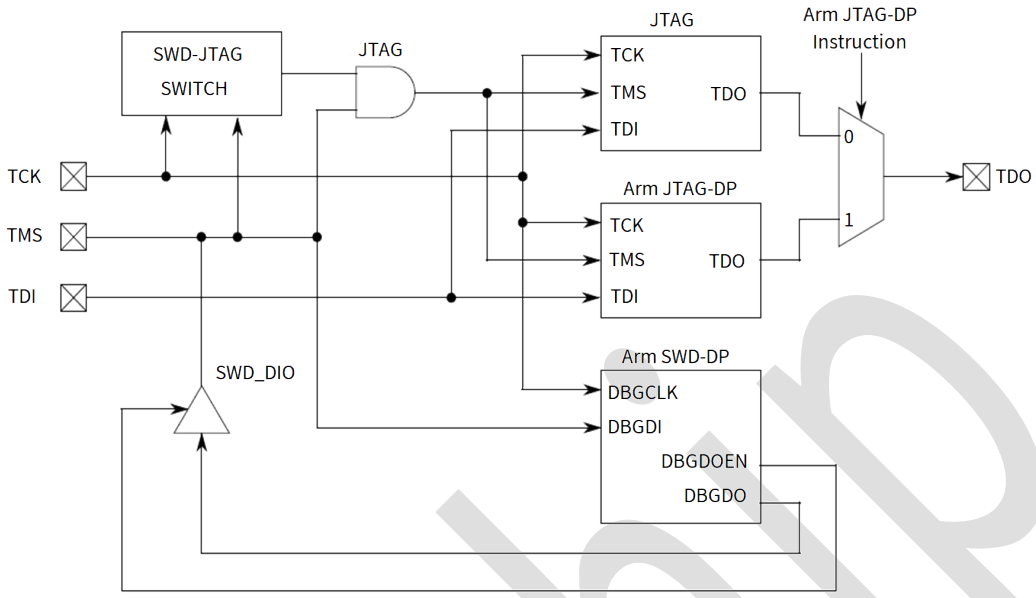

The following figure shows the FC7240F2M system-level ARM CPU debug structure.

**Figure 5. System-level CPU debug architecture**



The below diagram shows the JTAG controller and ARM JTAG Debug Port (DP)/SWD DP connection.

**Figure 6. JTAG/SWD connection diagram**




## 5.1 Debug Pin Descriptions

After Power-on Reset (POR), all debug port pins default to their JTAG functionality and can later be reassigned to their alternate functionalities. In SWD mode, JTAG_TDI can be configured to alternate GPIO functions.

**Table 11. JTAG/SWD pin configuration**

| Pin Name | JTAG Debug Port | | SWD Debug Port | | Internal Pull-up/down |
|---|---|---|---|---|---|
| | Type | Description | Type | Description | |
| JTAG_TMS/ SWD_DIO | I/O | JTAG Test Mode Selection | I/O | Serial Wire Data | Pull-up |
| JTAG_TCLK/ SWD_CLK | I | JTAG Test Clock | I | Serial Wire Clock | Pull-down |
| JTAG_TDI | I | JTAG Test Data Input | - | - | Pull-up |
| JTAG_TDO | O | JTAG Test Data Output | O | Trace output over a single pin | N/C |

## 5.2 JTAG TAP FSM

The following describes different states of the Test Access Port (TAP) controller:

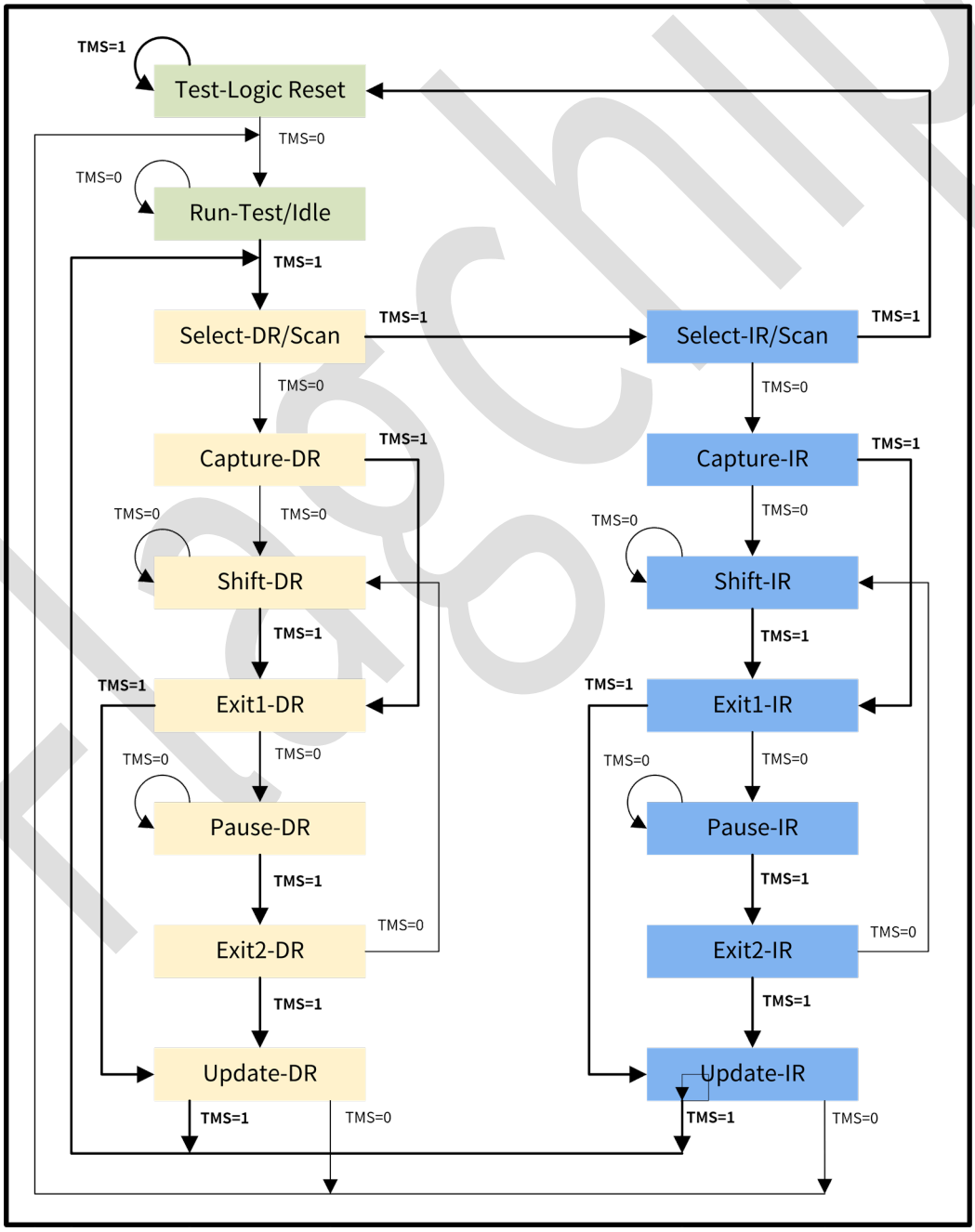
- **Test-Logic-Reset:** It resets the JTAG circuits. In whatever state the TAP controller may be at, it will go back to this state if TMS is set to 1 for 5 consecutive TCK cycles.
- **Run-Test/Idle**: This is a state in which the Finite State Machine (FSM) is waiting for some test operations to complete.
- **Select-DR/Scan and Select-IR/Scan**: This is a temporary state that allows the test data sequence for the corresponding Register (the IR in Select-IR/Scan state and the selected DR in Select-DR/Scan state) to be initiated.
- **Capture-DR and Capture-IR**: In this state, data can be loaded in parallel to the corresponding Register.
- **Shift-DR and Shift-IR**: In this state, the required test data is loaded (or unloaded) serially into (or from) the corresponding Register. If you refer the figure below, when the TAP controller is in this state, it will stay at this state

as long as TMS = 0. For each clock cycle, one data bit is shifted into (or out of) the selected Register through TDI (or TDO).

- **Exit1-DR and Exit1-IR:** All parallel-loaded (from the Capture-DR and Capture-IR state) or serial-loaded (from the Shift-DR and Shift IR state) data are held in the Register in this state.
- **Pause-DR and Pause-IR:** The FSM pauses its function here to wait for some external operation.
- **Exit2-DR and Exit2-IR:** This state represents the end of the Pause-DR or Pause-IR operation, and allows the TAP controller to go back to Shift-DR or Shift-IR state for more data to be shifted in (or shifted out).
- **Update-DR and Update-IR:** The test data stored in the first flop of Register (typically all the Registers have two flops for each bit, we will discuss it later) is loaded to the second flop in this state.

The below figure shows the JTAG FSM transition diagram.

**Figure 7. State transition diagram of TAP Controller FSM**



## 5.3 TAP Connection

The chip Test Access Port (TAP) is connected in parallel with Arm Test Access Port (TAP) controller. The chip TAP Instruction Register (IR) codes overlay Arm JTAG controller IR codes without conflict. For the available IR codes, refer to the IR codes table below. The outputs of the TAPs (TDO) are muxed based on the selected TAP. After power on reset, Arm's IDCODE (IR = 1110b) is selected.

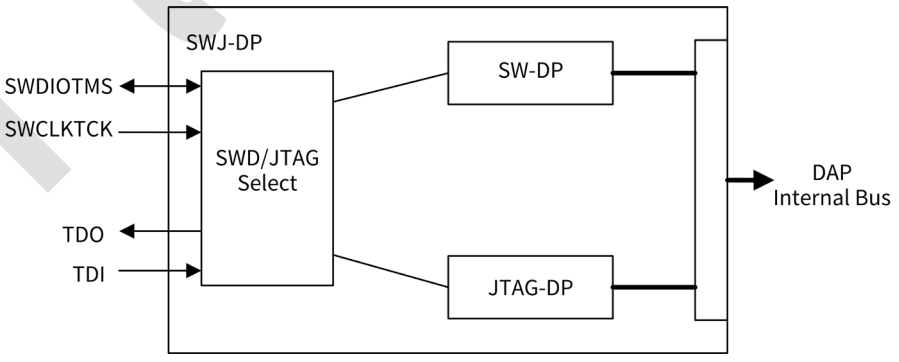
**Table 12. IR code assignments**

| Instruction | IR code | Instruction Purpose |
|---|---|---|
| IDCODE | 0000 | Selects device identification register for shifting |
| SAMPLE/PRELOAD | 0010 | Selects boundary scan register for shifting, sampling, and preloading without disturbing the functional operation |
| SAMPLE | 0011 | Selects boundary scan register for shifting and sampling without disturbing the functional operation |
| EXTEST | 0100 | Selects boundary scan register while applying preloaded values to output pins and asserting functional reset |
| HIGHZ | 1001 | Selects bypass register while three-stating all output pins and asserting functional reset |
| CLAMP | 1100 | Selects bypass register while applying preloaded values to output pins and asserting functional reset |
| ARM_IDCODE | 1110 | ARM JTAG-DP Instruction |
| BYPASS | 1111 | Selects bypass register for data operations |
| ARM JTAG-DP reserved | 1000, 1010, 1011, 1100 | These instructions will go to the Arm JTAG-DP controller. Please see the Arm JTAG-DP documentation for more information on these instructions |
| Reserved | All others | Decoded to select bypass register |

## 5.4 ARM AP and DP Connection

The FC7240F2M supports the Serial Wire/JTAG Debug Port (SWJ-DP), which supports both JTAG-DP and SW-DP. The following shows the SW-DP and JTAG-DP connections.

**Figure 8. SWJ-DP conceptual model**

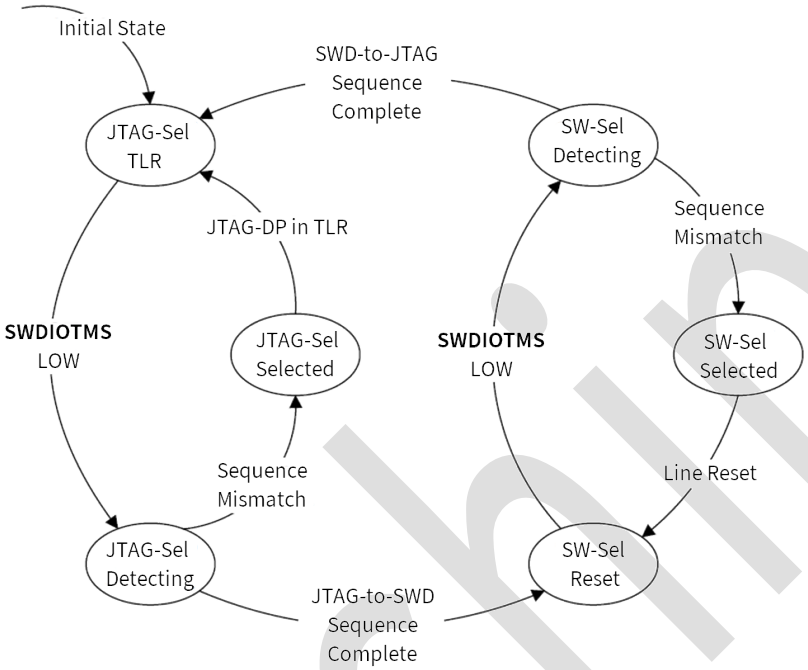



Flagchip

### 5.4.1 JTAG and SWD Switching Sequence

The below figure shows the JTAG and SWD switching FSM.

**Figure 9. JTAG and SWD switching FSM**




**For JTAG to SWD switching:**

1) Send more than 50 TCK cycles with TMS (SWD_DIO) = 1.
2) Send the 16-bit sequence on TMS (SWD_DIO) = 0111_1001_1110_0111 (MSB transmitted first).
3) Send more than 50 TCK cycles with TMS (SWD_DIO) = 1.

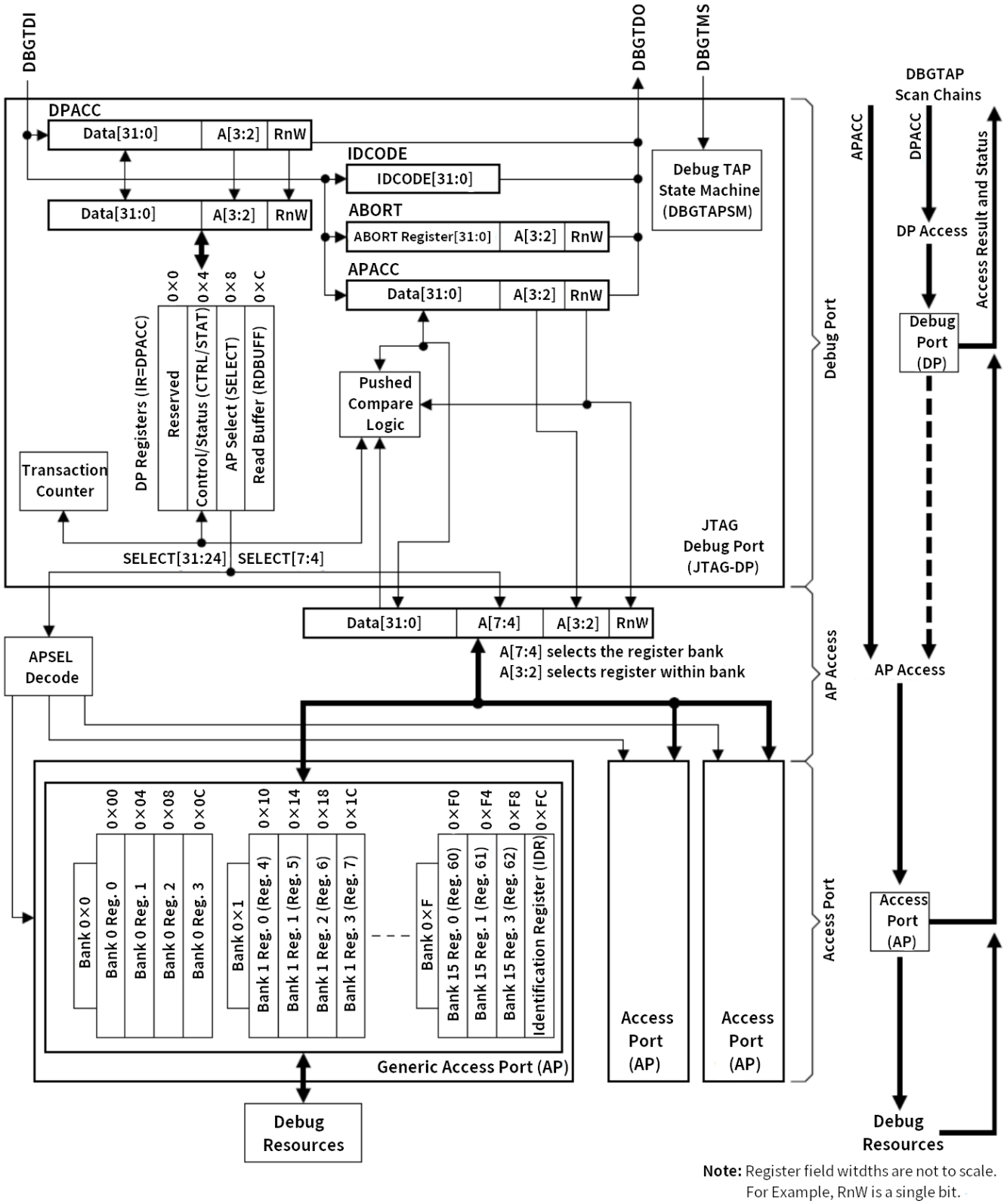
**For SWD to JTAG switching:**

1) Send more than 50 TCK cycles with TMS (SWD_DIO) = 1.
2) Send the 16-bit sequence on TMS (SWD_DIO) = 0011_1100_1110_0111 (MSB transmitted first).
3) Send more than 5 TCK cycles with TMS (SWD_DIO) = 1.

### 5.4.2 JTAG DP and AP Connection

For the ARM JTAG-DP and SW-DP related register mapping, please refer to ARM debug interface ADI document. The following figure shows the ARM JTAG DP and AP connection.

**Figure 10. Debug AP and DP connection diagram**




**Note:** Register field witdths are not to scale. For Example, RnW is a single bit.

APSEL, SELECT [31:24] = 0x00, select ARM CM7_0 AHB AP

See ARM documents for detailed AHB-AP information.

APSEL, SELECT [31:24] = 0x0F, select the SYS AP,

APBANKSEL, SELECT [7:4] = 0x00, select the SYS AP Status and Ctrl register bank,

A [3:2] = 2'b00, select Status register

A [3:2] = 2'b01, select Control register

A [3:2] = 2'b10, select Core Status 0 register

A [3:2] = 2'b11, select Core Status 1 register

APBANKSEL, SELECT [7:4] = 0x01, select the SYS AP Mail box register bank,

A [3:2] = 2'b00, select MDI register

A [3:2] = 2'b01, select MDO register

APBANKSEL, SELECT [7:4] = 0x02, select the SYS AP Status1/2 register bank,

A [3:2] = 2'b00, select Status1 register

A [3:2] = 2'b01, select Status2 register

APBANKSEL, SELECT [7:4] = 0x0F, select the SYS AP ID register bank,

A [3:2] = 2'b11, select ID register

## 5.5 SYS AP

Except for the ARM AHB-AP, the FC7240F2M has implemented SYS-AP.

The SYS-AP is accessible as Debug Access Port 15 with the available registers shown in the table below. SYS-AP control and status registers are used for debug purposes or control such as flash mass erase.

**Table 13. SYS AP register**

| SELECT(APSEL) | Register | Description |
|---|---|---|
| 0x0F00_0000 | Status | See SYS AP Status register |
| 0x0F00_0004 | Control | See SYS AP Control register |
| 0x0F00_0008 | Core Status0 | See SYS AP Core Status0 register |
| 0x0F00_000C | Core Status1 | See SYS AP Core Status1 register |
| 0x0F00_0010 | Debug Mailbox Data IN | SYS AP MDI register; general 32bit register; input from debug to device |
| 0x0F00_0014 | Debug Mailbox Data OUT | SYS AP MDO register; general 32bit register; output from device to debug |
| 0x0F00_0020 | Status1 | SYS AP Status1 register |
| 0x0F00_0024 | Status2 | System Debug Register, refer to CTRL bits for detailed registers |
| 0x0F00_00FC | ID | Read-only identification register; always read as 0x00FC_0010 |

The two tables below list the SYS AP control and status register bit definition. For the ID register, it is always read back as 0x00FC_0010.

**Table 14. SYS AP status register**

| Bit | Name | Description |
|---|---|---|
| 0 | Flash Mass Erase Complete | The bit is cleared at the launch of a mass erase operation by writing the SYS AP Control Register bit 0 (flash mass erase in progress). This bit is set after the flash control has completed the mass erase operation. If the mass erase is disabled (via SEC settings), the bit will not be set. |
| 1 | Flash Memory Ready | Indicates flash initialization is done; debug can access MCU through the debug port. |
| 2 | System Security | Indicates the system security state; 0 means ARM AHB-AP bus access is allowed. |
| 3 | System Reset | Indicates the system reset status.<br>0: System is in reset<br>1: System is not in reset |
| 4 | Reserved | Reserved |
| 5 | Flash Mass Erase Allow | Indicates if flash mass erase is allowed. |
| 6-9 | Reserved | Reserved |
| 10-16 | Reserved | Reserved |
| 17 | System Wait | Indicates if the system enters wait mode. |
| 18 | System Stop/Standby | Indicates the system enters stop or standby mode. |
| 19-23 | Reserved | Reserved |
| 24 | New Debug Mailbox Data Output Available | 1 means new Debug Mailbox Data Output (MDO) is available for debug to read; debug read MDO will automatically clear the bit. |
| 25-31 | Reserved | Reserved |

**Table 15. SYS AP control register**

| Bit | Name | Description |
|---|---|---|
| 0 | Flash Mass Erase in Progress | Set to trigger mass erase. Cleared by hardware after mass erase is done. If mass erase is disabled (via SEC settings), the erase request does not occur, and the bit keeps asserted until the next system reset. |
| 1 | Debug Disable | Set to disable the debug; clear to allow debug operation. |
| 2-9 | CPU Debug Request | Set to force core to halt. If the core is in stop mode, this bit can be used to wake up the core and transition to the halt state.<br>2: CPU0<br>3-9: Reserved |
| 10 | Reserved | Reserved |
| 11 | CPU Debug Restart | Set to force all CPU debug restarted, CPU to exit the halt state. |
| 12 | System Reset Request | Set to force a system reset. Clear the bit to release system from reset. |
| 13 | CPU0 Hold in Reset | Control if hold CPU0 in reset at the end of reset sequence. |
| 14 | CPU0 Hold in Wait | Control if hold CPU0 in wait at the end of reset sequence. |
| 15 | Reserved | Reserved |

**Table 15. SYS AP control register (continued)**

| Bit | Name | Description |
|---|---|---|
| 16 | Reserved | Reserved |
| 17 | New Mailbox Input Available | 1 means new Mailbox Data Input (MDI) is available; device read MDI register will automatically clear the bit. |
| 18-20 | Status2 Register Selection | 000: Select UID0<br>001: Select UID1<br>010: Select UID2<br>011: Select UID3<br>100: NVR version<br>101-111: Reserved<br>Can be written even system secure |
| 21-31 | Reserved | Reserved |

***Note:*** If system is secured, only allow write to Control register bits 0/12/17/18/19/20 and MDI registers. Other registers and register bits are not allowed to be written.

**Table 16. Core Status0 register**

| Bit | Name | Description |
|---|---|---|
| 0-7 | CPU in Reset | Indicates which core is in reset state.<br>0: CPU0 in reset state<br>1-7: Reserved |
| 8-15 | Core Debug Status | Indicates which core enters the halt state.<br>8: CPU0<br>9-15: Reserved |
| 16-23 | CPU Wait | Indicates which CPU enters the SLEEP mode.<br>16: CPU0<br>17-23: Reserved |
| 24-31 | CPU Stop | Indicate which CPU enters the DEEP SLEEP mode.<br>24: CPU0<br>25-31: Reserved |

**Table 17. Core Status1 register**

| Bit | Name | Description |
|---|---|---|
| 0-7 | CPU Debug Restarted | Indicates which CPU has returned to Normal mode from Debug mode.<br>0: CPU0<br>1-7: Reserved |
| 8-31 | Reserved | Reserved |

**Table 18. SYS_AP Status1 register**

| Bit | Name | Description |
|---|---|---|
| 0 | RESET 1->0 switching | Indicates system reset has 1->0 switching change, write 1 to clear (even system in secure state) |
| 1 | RESET 0->1 switching | Indicates system reset has 0->1 switching change, write 1 to clear (even system in secure state) |
| 2 | OTA0 Active Bank Status | Indicates OTA0 active bank |
| 3-4 | Reserved | Reserved |
| 5 | BOOT ROM Access Disable Status | Indicates current boot ROM register access status |
| 6 | Security Boot Status | Indicates current secure boot enable or disable |
| 7 | Boot from Flash Status | Indicates current boot from flash or ROM |
| 8 | Reserved | Reserved |
| 9 | Reserved | Reserved |
| 10 | OEM_DEV | Indicates lifecycle status (secure boot only) |
| 11 | OEM_PDT | Indicates lifecycle status (secure boot only) |
| 12 | IN_FIELD | Indicates lifecycle status (secure boot only) |
| 13 | PRE_FA | Indicates lifecycle status (secure boot only) |
| 14 | FA | Indicates lifecycle status (secure boot only) |
| 15 | Debug Re-Enable Allow Status | Debug Re-enable Allow Status (DEA) |
| 16 | SEC Register Write Lock Status | SEC Register Write Lock Status (RWL) |
| 17 | NVR Load Status | Indicates any NVR load error |
| 18 | Reserved | Reserved |
| 19 | OTA Load Status | Indicates any OTA load error |
| 20 | Flash Control Load Status | Indicates any Flash control load error |
| 21 | NVR Key Load Status | Indicates any NVR key load error |
| 22 | HSM Load Status | Indicates any HSM load error |
| 23 | Lifecycle Load Status | Indicates any Lifecycle load error |
| 24 | FlexCore Mode | Indicates current FlexCore mode status |
| 25 | Flash 128-bit Mode | Indicates current flash 128-bit mode status |

Flagchip

# Chapter 6 System Block Overview

## 6.1 Direct Memory Access (DMA)

- Support up to 16 DMA channels
- Support up to 64 DMA sources for each channel
- Support lockstep monitor for DMA
- Support address and data ECC for DMA CFG RAM

## 6.2 CPU System Control (CSC) Module

- CM7 has a dedicated CSC module
- CSC provides the IP low power handshake control
- CSC provides the corresponding CM7 clock control status and software interrupt generation
- CSC0 also contains the end-to-end ECC error related control, the system low power clock control and standby mode entry sequence control, and CMU3 reset control
- CSC resides in Standby domain

## 6.3 System Control Module (SCM)

The SCM contains below controls:

- End-to-end ECC enable control on the matrix etc.
- ECC enable control on CM7 ITCM/DTCM
- CM7 initial vectors (loaded from the non-volatile region (NVR))
- CM7 core hold control
- Internal connection routing control on ADC/FCSPI/FCUART/FTU/MSC/FLEXCAN/TPU etc.
- Sub-system low-power status information of all CM7 cores
- HSM interface/controls, which used to communicate between CM7 hosts and HSM
- CHIP ID related information

## 6.4 CPU Private Module (CPM)

- Each core contains one private CPM on External Private Peripheral Bus (EPPB)
- CPM contains the FPU related interrupt handle

## 6.5 AHB Overlay

- Each core contains 3 overlay regions that can map three PFLASH regions to the system RAM address.
- Each core contains 1 overlay region that can map a virtual flash region to a PFLASH region.

## 6.6 Mailbox (MB)

- Support 4 communication channels and 2 interrupt channels
- Each communication channel supports semaphore capability and 2 × 32-bit data registers
- Each communication channel can send an interrupt request to 2 interrupt channels
- Interrupt channel 0 assigned to CM7; interrupt channel 1 assigned to HSM

Flagchip

## 6.7 Matrix Access Monitor (MAM)

- Each MAM contains the WDOG monitor for each master access port. If access is not finished in 512*16 bus clocks, it will generate an MAM timeout error. If the user enables the flash stall function while the flash is under erase/program, it may trigger time out. In this case, it is better to disable the WDOG.
- Each slave port contains the block access protection. The PFlash is 128 KB per block, DFlash is 8 KB per block, AFCB is 4 KB per block, and SRAM is 16 KB per block (comprises ITCM/DTCM and System RAM0/1/2).

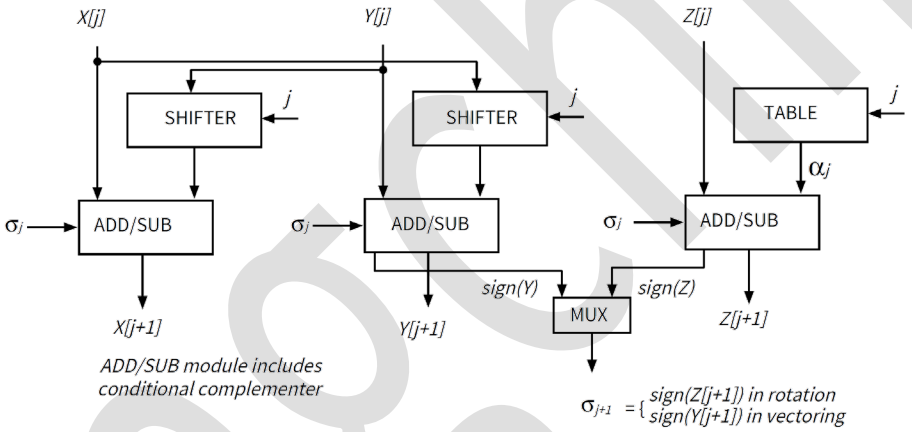
## 6.8 CORDIC

The FC7240F2M supports the CORDIC Accelerator to accelerate the mathematic operation including Sin/ Cos/ Sinh/ Cosh/ Square. The accelerator supports fixed point input format.

The FC7240F2M also supports different input limit ranges and extra scale functions (compared with the FC7300 series). Refer to Chapter *CORDIC* for details.

The below figure shows one CORDIC micro-operation. Each iteration contains 2 micro-operations.

**Figure 11. CORDIC micro-operation diagram**




## 6.9 Security Controller (SEC)

- Support both secure and non-secure boot
- Support the lifecycle related control
- Support the NVR configuration load
- Support controls such as the debug enable/mass erase enable

## 6.10 Wake-up Unit (WKU)

- Support configurable wakeup capability

Below are the WKU input channel assignments.

**Table 19. WKU input channel connection**

| WKU Inputs | Wake-up Sources |
|---|---|
| WKU_INPUT0 | Reserved |
| WKU_INPUT1 | Reserved |
| WKU_INPUT2 | Reserved |
| WKU_INPUT3 | Reserved |
| WKU_INPUT4 | Reserved |
| WKU_INPUT5 | Reserved |

**Table 19. WKU input channel connection (continued)**

| WKU Inputs | Wake-up Sources |
|---|---|
| WKU_INPUT6 | FCSPI0 |
| WKU_INPUT7 | FCUART0 |
| WKU_INPUT8 | Reserved |
| WKU_INPUT9 | Reserved |
| WKU_INPUT10 | CMP0 |
| WKU_INPUT11 | CMP1 |
| WKU_INPUT12 | Reserved |
| WKU_INPUT13 | TSTMP0 |
| WKU_INPUT14 | RTC Alarm |
| WKU_INPUT15 | RTC Seconds |
| WKU_INPUT16 | FCPIT0 |
| WKU_INPUT17 | CMU0 |
| WKU_INPUT18 | AONTIMER0 |
| WKU_INPUT19 | GPIOA |
| WKU_INPUT20 | Reserved |
| WKU_INPUT21 | GPIOC |
| WKU_INPUT22 | GPIOD |
| WKU_INPUT23 | GPIOE |

## 6.11 Flash Memory Controller (FMC)

- Support the pre-fetch function and read stall function
- Support the mini-linebuffer function
- Support cache with ECC protection
- Support OTA function

## 6.12 Flash Control

- Support the Flash program/erase control
- Support the Flash initialization control
- Support the Flash parallel read capability
- Support ECC protection on both address and data
- For parallel mode, the PFlash sector size is 4 KB, and the minimum program size is 16 bytes
- For non-parallel mode, the PFlash sector size is 2 KB, and the minimum program size is 8 bytes

## 6.13 On-Chip Connection

### 6.13.1 Logic Unit (LU)

- Support one LU
- Each LU supports functions including AND/OR/INVERT (AOI)/JK/RS/FF/LATCH

### 6.13.2 Trigger Select (TRGSEL) Sources

The FC7240F2M has 6 TRGSELs. The TRGSEL sources are listed in the following table.

#### 6.13.2.1 TRGSEL0

**Table 20. TRGSEL0 sources**

| TRGSEL0 | | | |
|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** |
| 0 | LU0_INPUT0A | 0 | VSS |
| 1 | LU0_INPUT0B | 1 | VDD |
| 2 | LU0_INPUT0C | 2 | SCM_SW_TRG0 |
| 3 | LU0_INPUT0D | 3 | SCM_SW_TRG1 |
| 4 | LU0_INPUT1A | 4 | SCM_SW_TRG2 |
| 5 | LU0_INPUT1B | 5 | SCM_SW_TRG3 |
| 6 | LU0_INPUT1C | 6 | TRGSEL_IN0 |
| 7 | LU0_INPUT1D | 7 | TRGSEL_IN1 |
| 8 | LU0_INPUT2A | 8 | TRGSEL_IN2 |
| 9 | LU0_INPUT2B | 9 | TRGSEL_IN3 |
| 10 | LU0_INPUT2C | 10 | TRGSEL_IN4 |
| 11 | LU0_INPUT2D | 11 | TRGSEL_IN5 |
| 12 | LU0_INPUT3A | 12 | TRGSEL_IN6 |
| 13 | LU0_INPUT3B | 13 | TRGSEL_IN7 |
| 14 | LU0_INPUT3C | 14 | TRGSEL_IN8 |
| 15 | LU0_INPUT3D | 15 | TRGSEL_IN9 |
| 16 | TRGSEL1_INPUT0 | 16 | TRGSEL_IN10 |
| 17 | TRGSEL1_INPUT1 | 17 | TRGSEL_IN11 |
| 18 | TRGSEL1_INPUT2 | 18 | TRGSEL_IN12 |
| 19 | TRGSEL1_INPUT3 | 19 | TRGSEL_IN13 |
| 20 | TRGSEL1_INPUT4 | 20 | TRGSEL_IN14 |
| 21 | TRGSEL1_INPUT5 | 21 | TRGSEL_IN15 |
| 22 | TRGSEL1_INPUT6 | 22 | CMP0_OUT |
| 23 | TRGSEL1_INPUT7 | 23 | CMP1_OUT |
| 24 | FTU0_TRG0 | 24 | FCPIT0_CH0 |
| 25 | FTU1_TRG0 | 25 | FCPIT0_CH1 |
| 26 | FTU2_TRG0 | 26 | FCPIT0_CH2 |
| 27 | FTU3_TRG0 | 27 | FCPIT0_CH3 |
| 28 | FTU4_TRG0 | 28 | AONTIMER0_TRGO |
| 29 | FTU5_TRG0 | 29 | FTU0_RELOAD_TRG |
| 30 | FTU6_TRG0 | 30 | FTU0_MATCH_TRG |
| 31 | FTU7_TRG0 | 31 | FTU1_RELOAD_TRG |
| 32 | SENT0_CH0TRG | 32 | FTU1_MATCH_TRG |
| 33 | SENT0_CH1TRG | 33 | FTU2_RELOAD_TRG |
| 34 | SENT0_CH2TRG | 34 | FTU2_MATCH_TRG |
| 35 | SENT0_CH3TRG | 35 | FTU3_RELOAD_TRG |
| 36 | CMP0_SAMPLE_EN | 36 | FTU3_MATCH_TRG |
| 37 | CMP1_SAMPLE_EN | 37 | FTU4_RELOAD_TRG |

Flagchip

**Table 20. TRGSEL0 sources (continued)**

| TRGSEL0 | | | |
|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** |
| 38 | | 38 | FTU4_MATCH_TRG |
| 39 | | 39 | FTU5_RELOAD_TRG |
| 40 | | 40 | FTU5_MATCH_TRG |
| 41 | | 41 | FTU6_RELOAD_TRG |
| 42 | | 42 | FTU6_MATCH_TRG |
| 43 | | 43 | FTU7_RELOAD_TRG |
| 44 | | 44 | FTU7_MATCH_TRG |
| 45 | | 45 | LU0_OUT0A |
| 46 | | 46 | LU0_OUT1A |
| 47 | | 47 | LU0_OUT2A |
| 48 | | 48 | LU0_OUT3A |
| 49 | | 49 | LU0_OUT0B |
| 50 | | 50 | LU0_OUT1B |
| 51 | | 51 | LU0_OUT2B |
| 52 | | 52 | LU0_OUT3B |
| 53 | | 53 | PTIMER0_ADCCH0_TRG |
| 54 | | 54 | PTIMER0_PULSE |
| 55 | | 55 | ADC0_EOSG[0] |
| 56 | | 56 | ADC0_EOSG[1] |
| 57 | | 57 | ADC0_EOSG[2] |
| 58 | | 58 | ADC0_EOSG[3] |
| 59 | | 59 | PTIMER1_ADCCH0_TRG |
| 60 | | 60 | PTIMER1_PULSE |
| 61 | | 61 | ADC1_EOSG[0] |
| 62 | | 62 | ADC1_EOSG[1] |
| 63 | | 63 | ADC1_EOSG[2] |
| 64 | | 64 | ADC1_EOSG[3] |
| 65 | | 65 | RTC_ALARM |
| 66 | | 66 | RTC_SECOND |
| 67 | | 67 | GPIOA TRG |
| 68 | | 68 | GPIOB TRG |
| 69 | | 69 | GPIOC TRG |
| 70 | | 70 | GPIOD TRG |
| 71 | | 71 | GPIOE TRG |
| 72 | | 72 | FTU0_CH0_OUT |
| 73 | | 73 | FTU0_CH1_OUT |
| 74 | | 74 | FTU0_CH2_OUT |
| 75 | | 75 | FTU0_CH3_OUT |

Flagchip

**Table 20. TRGSEL0 sources (continued)**

| TRGSEL0 | | | |
|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** |
| 76 | | 76 | FTU0_CH4_OUT |
| 77 | | 77 | FTU0_CH5_OUT |
| 78 | | 78 | FTU0_CH6_OUT |
| 79 | | 79 | FTU0_CH7_OUT |
| 80 | | 80 | FTU3_CH0_OUT |
| 81 | | 81 | FTU3_CH1_OUT |
| 82 | | 82 | FTU3_CH2_OUT |
| 83 | | 83 | FTU3_CH3_OUT |
| 84 | | 84 | FTU3_CH4_OUT |
| 85 | | 85 | FTU3_CH5_OUT |
| 86 | | 86 | FTU3_CH6_OUT |
| 87 | | 87 | FTU3_CH7_OUT |
| 88 | | 88 | FTU6_CH0_OUT |
| 89 | | 89 | FTU6_CH1_OUT |
| 90 | | 90 | FTU6_CH2_OUT |
| 91 | | 91 | FTU6_CH3_OUT |
| 92 | | 92 | FTU6_CH4_OUT |
| 93 | | 93 | FTU6_CH5_OUT |
| 94 | | 94 | FTU6_CH6_OUT |
| 95 | | 95 | FTU6_CH7_OUT |
| 96 | | 96 | TPU_TRG0 |
| 97 | | 97 | TPU_TRG1 |
| 98 | | 98 | TPU_TRG2 |
| 99 | | 99 | TPU_TRG3 |
| 100 | | 100 | TPU_TRG4 |
| 101 | | 101 | TPU_TRG5 |
| 102 | | 102 | TPU_TRG6 |
| 103 | | 103 | TPU_TRG7 |
| 104 | | 104 | TPU_TRG8 |
| 105 | | 105 | TPU_TRG9 |
| 106 | | 106 | TPU_TRG10 |
| 107 | | 107 | TPU_TRG11 |
| 108 | | 108 | TPU_TRG12 |
| 109 | | 109 | TPU_TRG13 |
| 110 | | 110 | TPU_TRG14 |
| 111 | | 111 | TPU_TRG15 |
| 112 | | 112 | TPU_TRG16 |
| 113 | | 113 | TPU_TRG17 |

**Table 20. TRGSEL0 sources (continued)**

| TRGSEL0 | | | |
|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** |
| 114 | | 114 | TPU_TRG18 |
| 115 | | 115 | TPU_TRG19 |
| 116 | | 116 | TPU_TRG20 |
| 117 | | 117 | TPU_TRG21 |
| 118 | | 118 | TPU_TRG22 |
| 119 | | 119 | TPU_TRG23 |
| 120 | | 120 | TPU_TRG24 |
| 121 | | 121 | TPU_TRG25 |
| 122 | | 122 | TPU_TRG26 |
| 123 | | 123 | TPU_TRG27 |
| 124 | | 124 | TPU_TRG28 |
| | | 125 | TPU_TRG29 |
| | | 126 | TPU_TRG30 |
| | | 127 | TPU_TRG31 |

#### *6.13.2.2 TRGSEL1*

**Table 21. TRGSEL1 sources**

| TRGSEL1 | | | |
|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** |
| 0 | DMA_CH0 | 0 | VSS |
| 1 | DMA_CH1 | 1 | VDD |
| 2 | DMA_CH2 | 2 | SCM_SW_TRG4 |
| 3 | DMA_CH3 | 3 | SCM_SW_TRG5 |
| 4 | TRGSEL_OUT0 | 4 | SCM_SW_TRG6 |
| 5 | TRGSEL_OUT1 | 5 | SCM_SW_TRG7 |
| 6 | TRGSEL_OUT2 | 6 | TRGSEL_IN0 |
| 7 | TRGSEL_OUT3 | 7 | TRGSEL_IN1 |
| 8 | TRGSEL_OUT4 | 8 | TRGSEL_IN2 |
| 9 | TRGSEL_OUT5 | 9 | TRGSEL_IN3 |
| 10 | TRGSEL_OUT6 | 10 | TRGSEL_IN4 |
| 11 | TRGSEL_OUT7 | 11 | TRGSEL_IN5 |
| 12 | FCUART0_TRGI | 12 | TRGSEL_IN6 |
| 13 | FCUART1_TRGI | 13 | TRGSEL_IN7 |
| 14 | FCUART2_TRGI | 14 | TRGSEL_IN8 |
| 15 | FCUART3_TRGI | 15 | TRGSEL_IN9 |
| 16 | FCUART4_TRGI | 16 | TRGSEL_IN10 |
| 17 | FCUART5_TRGI | 17 | TRGSEL_IN11 |

**Table 21. TRGSEL1 sources (continued)**

| TRGSEL1 | | | |
|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** |
| 18 | FCUART6_TRGI | 18 | TRGSEL_IN12 |
| 19 | FCUART7_TRGI | 19 | TRGSEL_IN13 |
| 20 | FCSPI0_TRGI | 20 | TRGSEL_IN14 |
| 21 | FCSPI1_TRGI | 21 | TRGSEL_IN15 |
| 22 | FCSPI2_TRGI | 22 | CMP0_OUT |
| 23 | FCSPI3_TRGI | 23 | CMP1_OUT |
| 24 | FCSPI4_TRGI | 24 | FCPIT0_CH0 |
| 25 | FCSPI5_TRGI | 25 | FCPIT0_CH1 |
| 26 | FCIIC0_TRGI | 26 | FCPIT0_CH2 |
| 27 | FCIIC1_TRGI | 27 | FCPIT0_CH3 |
| 28 | | 28 | AONTIMER0_TRGO |
| 29 | | 29 | LU0_OUT0A |
| 30 | | 30 | LU0_OUT1A |
| 31 | | 31 | LU0_OUT2A |
| 32 | | 32 | LU0_OUT3A |
| 33 | | 33 | LU0_OUT0B |
| 34 | | 34 | LU0_OUT1B |
| 35 | | 35 | LU0_OUT2B |
| 36 | | 36 | LU0_OUT3B |
| 37 | | 37 | RTC_ALARM |
| 38 | | 38 | RTC_SECOND |
| 39 | | 39 | GPIOA TRG |
| 40 | | 40 | GPIOB TRG |
| 41 | | 41 | GPIOC TRG |
| 42 | | 42 | GPIOD TRG |
| 43 | | 43 | GPIOE TRG |
| 44 | | 44 | TRGSEL1_INPUT0 |
| 45 | | 45 | TRGSEL1_INPUT1 |
| 46 | | 46 | TRGSEL1_INPUT2 |
| 47 | | 47 | TRGSEL1_INPUT3 |
| 48 | | 48 | TRGSEL1_INPUT4 |
| 49 | | 49 | TRGSEL1_INPUT5 |
| 50 | | 50 | TRGSEL1_INPUT6 |
| 51 | | 51 | TRGSEL1_INPUT7 |
| 52 | | 52 | FCIIC0_MTRG |
| 53 | | 53 | FCIIC0_STRG |
| 54 | | 54 | FCIIC1_MTRG |
| 55 | | 55 | FCIIC1_STRG |

**Table 21. TRGSEL1 sources (continued)**

| TRGSEL1 | | | |
|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** |
| 56 | | 56 | FCSPI0_FRAME |
| 57 | | 57 | FCSPI0_RX_DONE |
| 58 | | 58 | FCSPI1_FRAME |
| 59 | | 59 | FCSPI1_RX_DONE |
| 60 | | 60 | FCSPI2_FRAME |
| 61 | | 61 | FCSPI2_RX_DONE |
| 62 | | 62 | FCSPI3_FRAME |
| 63 | | 63 | FCSPI3_RX_DONE |
| 64 | | 64 | FCSPI4_FRAME |
| 65 | | 65 | FCSPI4_RX_DONE |
| 66 | | 66 | FCSPI5_FRAME |
| 67 | | 67 | FCSPI5_RX_DONE |
| 68 | | 68 | FCUART0_RX_DONE |
| 69 | | 69 | FCUART0_TX_DONE |
| 70 | | 70 | FCUART0_RX_IDLE |
| 71 | | 71 | FCUART1_RX_DONE |
| 72 | | 72 | FCUART1_TX_DONE |
| 73 | | 73 | FCUART1_RX_IDLE |
| 74 | | 74 | FCUART2_RX_DONE |
| 75 | | 75 | FCUART2_TX_DONE |
| 76 | | 76 | FCUART2_RX_IDLE |
| 77 | | 77 | FCUART3_RX_DONE |
| 78 | | 78 | FCUART3_TX_DONE |
| 79 | | 79 | FCUART3_RX_IDLE |
| 80 | | 80 | FCUART4_RX_DONE |
| 81 | | 81 | FCUART4_TX_DONE |
| 82 | | 82 | FCUART4_RX_IDLE |
| 83 | | 83 | FCUART5_RX_DONE |
| 84 | | 84 | FCUART5_TX_DONE |
| 85 | | 85 | FCUART5_RX_IDLE |
| 86 | | 86 | FCUART6_RX_DONE |
| 87 | | 87 | FCUART6_TX_DONE |
| 88 | | 88 | FCUART6_RX_IDLE |
| 89 | | 89 | FCUART7_RX_DONE |
| 90 | | 90 | FCUART7_TX_DONE |
| 91 | | 91 | FCUART7_RX_IDLE |
| 92 | | 92 | ADC0_CMP |
| 93 | | 93 | ADC1_CMP |

#### 6.13.2.3 TRGSEL2

**Table 22. TRGSEL2 sources**

| TRGSEL2 | | | | |
|---|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** | **Comment** |
| 0 | FTU0_FLT0 | 0 | VSS | |
| 1 | FTU0_FLT1 | 1 | VDD | |
| 2 | FTU1_FLT0 | 2 | TRGSEL_OUT0 | From TRGSEL1 output (not from pin) |
| 3 | FTU1_FLT1 | 3 | TRGSEL_OUT1 | From TRGSEL1 output (not from pin) |
| 4 | FTU2_FLT0 | 4 | TRGSEL_OUT2 | From TRGSEL1 output (not from pin) |
| 5 | FTU2_FLT1 | 5 | TRGSEL_OUT3 | From TRGSEL1 output (not from pin) |
| 6 | FTU3_FLT0 | 6 | TRGSEL_OUT4 | From TRGSEL1 output (not from pin) |
| 7 | FTU3_FLT1 | 7 | TRGSEL_OUT5 | From TRGSEL1 output (not from pin) |
| 8 | FTU4_FLT0 | 8 | TRGSEL_OUT6 | From TRGSEL1 output (not from pin) |
| 9 | FTU4_FLT1 | 9 | TRGSEL_OUT7 | From TRGSEL1 output (not from pin) |
| 10 | FTU5_FLT0 | 10 | FTU_FLT0 | |
| 11 | FTU5_FLT1 | 11 | FTU_FLT1 | |
| 12 | FTU6_FLT0 | 12 | FTU_FLT2 | |
| 13 | FTU6_FLT1 | 13 | FTU_FLT3 | |
| 14 | FTU7_FLT0 | 14 | FTU_FLT4 | |
| 15 | FTU7_FLT1 | 15 | FTU_FLT5 | |
| 16 | FTU0_TRG1 | 16 | FTU_FLT6 | |
| 17 | FTU1_TRG1 | 17 | FTU_FLT7 | |
| 18 | FTU2_TRG1 | 18 | FTU_FLT8 | |
| 19 | FTU3_TRG1 | 19 | FTU_FLT9 | |
| 20 | FTU4_TRG1 | 20 | FTU_FLT10 | |
| 21 | FTU5_TRG1 | 21 | FTU_FLT11 | |
| 22 | FTU6_TRG1 | 22 | FTU_FLT12 | |
| 23 | FTU7_TRG1 | 23 | FTU_FLT13 | |
| 24 | MSC0_INJ0 | 24 | FTU_FLT14 | |
| 25 | MSC0_INJ1 | 25 | FTU_FLT15 | |
| 26 | TPU_FLTIN0 | 26 | FTU_FLT16 | |
| 27 | TPU_FLTIN1 | 27 | FTU_FLT17 | |
| 28 | TPU_FLTIN2 | 28 | FTU_FLT18 | |
| 29 | TPU_FLTIN3 | 29 | FTU_FLT19 | |
| 30 | | 30 | FTU_FLT20 | |
| 31 | | 31 | FTU_FLT21 | |

#### 6.13.2.4 TRGSEL3

**Table 23. TRGSEL3 sources**

| TRGSEL3 | | | | |
|---|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** | **Comment** |
| 0 | ISM0_MON0 | 0 | VSS | |

**Table 23. TRGSEL3 sources (continued)**

| TRGSEL3 | | | | |
|---|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** | **Comment** |
| 1 | ISM0_MON1 | 1 | VDD | |
| 2 | ISM0_MON2 | 2 | TRGSEL_OUT0 | from TRGSEL1 output |
| 3 | ISM0_MON3 | 3 | TRGSEL_OUT1 | from TRGSEL1 output |
| 4 | ISM0_MON4 | 4 | TRGSEL_OUT2 | from TRGSEL1 output |
| 5 | ISM0_MON5 | 5 | TRGSEL_OUT3 | from TRGSEL1 output |
| 6 | ISM0_MON6 | 6 | TRGSEL_OUT4 | from TRGSEL1 output |
| 7 | ISM0_MON7 | 7 | TRGSEL_OUT5 | from TRGSEL1 output |
| 8 | ISM0_REF0 | 8 | TRGSEL_OUT6 | from TRGSEL1 output |
| 9 | ISM0_REF1 | 9 | TRGSEL_OUT7 | from TRGSEL1 output |
| 10 | ISM0_REF2 | 10 | TRGSEL1_INPUT0 | from TRGSEL0 output |
| 11 | ISM0_REF3 | 11 | TRGSEL1_INPUT1 | from TRGSEL0 output |
| 12 | ISM0_REF4 | 12 | TRGSEL1_INPUT2 | from TRGSEL0 output |
| 13 | ISM0_REF5 | 13 | TRGSEL1_INPUT3 | from TRGSEL0 output |
| 14 | ISM0_REF6 | 14 | TRGSEL1_INPUT4 | from TRGSEL0 output |
| 15 | ISM0_REF7 | 15 | TRGSEL1_INPUT5 | from TRGSEL0 output |
| 16 | | 16 | TRGSEL1_INPUT6 | from TRGSEL0 output |
| 17 | | 17 | TRGSEL1_INPUT7 | from TRGSEL0 output |
| 18 | | 18 | LU0_OUT0A | |
| 19 | | 19 | LU0_OUT1A | |
| 20 | | 20 | LU0_OUT2A | |
| 21 | | 21 | LU0_OUT3A | |
| 22 | | 22 | FTU_IN0 | FTU routing option 0 from 7 FTU fcf_ind group0 |
| 23 | | 23 | FTU_IN1 | FTU routing option 0 from 7 FTU fcf_ind group0 |
| 24 | | 24 | FTU_IN2 | FTU routing option 0 from 7 FTU fcf_ind group0 |
| 25 | | 25 | FTU_IN3 | FTU routing option 0 from 7 FTU fcf_ind group0 |
| 26 | | 26 | FTU_IN4 | FTU routing option 0 from 7 FTU fcf_ind group0 |
| 27 | | 27 | FTU_IN5 | FTU routing option 0 from 7 FTU fcf_ind group0 |
| 28 | | 28 | FTU_IN6 | FTU routing option 0 from 7 FTU fcf_ind group0 |
| 29 | | 29 | FTU_IN7 | FTU routing option 0 from 7 FTU fcf_ind group0 |
| 30 | | 30 | FTU_IN8 | FTU routing option 0 from 7 FTU fcf_ind group1 |

**Table 23. TRGSEL3 sources (continued)**

| TRGSEL3 | | | | |
|---|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** | **Comment** |
| 31 | | 31 | FTU_IN9 | FTU routing option 0 from 7 FTU fcf_ind group1 |
| 32 | | 32 | FTU_IN10 | FTU routing option 0 from 7 FTU fcf_ind group1 |
| 33 | | 33 | FTU_IN11 | FTU routing option 0 from 7 FTU fcf_ind group1 |
| 34 | | 34 | FTU_IN12 | FTU routing option 0 from 7 FTU fcf_ind group1 |
| 35 | | 35 | FTU_IN13 | FTU routing option 0 from 7 FTU fcf_ind group1 |
| 36 | | 36 | FTU_IN14 | FTU routing option 0 from 7 FTU fcf_ind group1 |
| 37 | | 37 | FTU_IN15 | FTU routing option 0 from 7 FTU fcf_ind group1 |
| 38 | | 38 | FTU_OUT0 | FTU routing option 0 from 7 FTU output group0 |
| 39 | | 39 | FTU_OUT1 | FTU routing option 0 from 7 FTU output group0 |
| 40 | | 40 | FTU_OUT2 | FTU routing option 0 from 7 FTU output group0 |
| 41 | | 41 | FTU_OUT3 | FTU routing option 0 from 7 FTU output group0 |
| 42 | | 42 | FTU_OUT4 | FTU routing option 0 from 7 FTU output group0 |
| 43 | | 43 | FTU_OUT5 | FTU routing option 0 from 7 FTU output group0 |
| 44 | | 44 | FTU_OUT6 | FTU routing option 0 from 7 FTU output group0 |
| 45 | | 45 | FTU_OUT7 | FTU routing option 0 from 7 FTU output group0 |
| 46 | | 46 | FTU_OUT8 | FTU routing option 0 from 7 FTU output group1 |
| 47 | | 47 | FTU_OUT9 | FTU routing option 0 from 7 FTU output group1 |
| 48 | | 48 | FTU_OUT10 | FTU routing option 0 from 7 FTU output group1 |
| 49 | | 49 | FTU_OUT11 | FTU routing option 0 from 7 FTU output group1 |
| 50 | | 50 | FTU_OUT12 | FTU routing option 0 from 7 FTU |

**Table 23. TRGSEL3 sources (continued)**

| TRGSEL3 | | | | |
|---|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** | **Comment** |
| | | | | output group1 |
| 51 | | 51 | FTU_OUT13 | FTU routing option 0 from 7 FTU output group1 |
| 52 | | 52 | FTU_OUT14 | FTU routing option 0 from 7 FTU output group1 |
| 53 | | 53 | FTU_OUT15 | FTU routing option 0 from 7 FTU output group1 |
| 54 | | 54 | TPU_CH0 | From TPU channel fcf_ind |
| 55 | | 55 | TPU_CH1 | From TPU channel fcf_ind |
| 56 | | 56 | TPU_CH2 | From TPU channel fcf_ind |
| 57 | | 57 | TPU_CH3 | From TPU channel fcf_ind |
| 58 | | 58 | TPU_CH4 | From TPU channel fcf_ind |
| 59 | | 59 | TPU_CH5 | From TPU channel fcf_ind |
| 60 | | 60 | TPU_CH6 | From TPU channel fcf_ind |
| 61 | | 61 | TPU_CH7 | From TPU channel fcf_ind |

#### *6.13.2.5 TRGSEL4*

**Table 24. TRGSEL4 sources**

| TRGSEL4 | | | |
|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** |
| 0 | ADC0_TRG[0] | 0 | VSS |
| 1 | ADC0_TRG[1] | 1 | VDD |
| 2 | ADC0_TRG[2] | 2 | SCM_SW_TRG0 |
| 3 | ADC0_TRG[3] | 3 | SCM_SW_TRG1 |
| 4 | ADC1_TRG[0] | 4 | SCM_SW_TRG2 |
| 5 | ADC1_TRG[1] | 5 | SCM_SW_TRG3 |
| 6 | ADC1_TRG[2] | 6 | TRGSEL_IN8 |
| 7 | ADC1_TRG[3] | 7 | TRGSEL_IN9 |
| 8 | PTIMER0_TRG | 8 | TRGSEL_IN10 |
| 9 | PTIMER1_TRG | 9 | TRGSEL_IN11 |
| 10 | | 10 | TRGSEL_IN12 |
| 11 | | 11 | TRGSEL_IN13 |
| 12 | | 12 | TRGSEL_IN14 |
| 13 | | 13 | TRGSEL_IN15 |
| 14 | | 14 | LU0_OUT0A |
| 15 | | 15 | LU0_OUT1A |
| 16 | | 16 | LU0_OUT2A |

**Table 24. TRGSEL4 sources (continued)**

| TRGSEL4 | | | |
|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** |
| 17 | | 17 | LU0_OUT3A |
| 18 | | 18 | LU0_OUT0B |
| 19 | | 19 | LU0_OUT1B |
| 20 | | 20 | LU0_OUT2B |
| 21 | | 21 | LU0_OUT3B |
| 22 | | 22 | FCPIT0_CH0 |
| 23 | | 23 | FCPIT0_CH1 |
| 24 | | 24 | FCPIT0_CH2 |
| 25 | | 25 | FCPIT0_CH3 |
| 26 | | 26 | TRGSEL1_INPUT0 |
| 27 | | 27 | TRGSEL1_INPUT1 |
| 28 | | 28 | TRGSEL1_INPUT2 |
| 29 | | 29 | TRGSEL1_INPUT3 |
| 30 | | 30 | TRGSEL1_INPUT4 |
| 31 | | 31 | TRGSEL1_INPUT5 |
| 32 | | 32 | TRGSEL1_INPUT6 |
| 33 | | 33 | TRGSEL1_INPUT7 |
| 34 | | 34 | FTU0_RELOAD_TRG |
| 35 | | 35 | FTU0_MATCH_TRG |
| 36 | | 36 | FTU1_RELOAD_TRG |
| 37 | | 37 | FTU1_MATCH_TRG |
| 38 | | 38 | FTU2_RELOAD_TRG |
| 39 | | 39 | FTU2_MATCH_TRG |
| 40 | | 40 | FTU3_RELOAD_TRG |
| 41 | | 41 | FTU3_MATCH_TRG |
| 42 | | 42 | FTU4_RELOAD_TRG |
| 43 | | 43 | FTU4_MATCH_TRG |
| 44 | | 44 | FTU5_RELOAD_TRG |
| 45 | | 45 | FTU5_MATCH_TRG |
| 46 | | 46 | FTU6_RELOAD_TRG |
| 47 | | 47 | FTU6_MATCH_TRG |
| 48 | | 48 | FTU7_RELOAD_TRG |
| 49 | | 49 | FTU7_MATCH_TRG |
| 50 | | 50 | FTU0_CH0_OUT |
| 51 | | 51 | FTU0_CH1_OUT |
| 52 | | 52 | FTU0_CH2_OUT |
| 53 | | 53 | FTU0_CH3_OUT |
| 54 | | 54 | FTU0_CH4_OUT |

**Table 24. TRGSEL4 sources (continued)**

| TRGSEL4 | | | |
|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** |
| 55 | | 55 | FTU0_CH5_OUT |
| 56 | | 56 | FTU0_CH6_OUT |
| 57 | | 57 | FTU0_CH7_OUT |
| 58 | | 58 | FTU3_CH0_OUT |
| 59 | | 59 | FTU3_CH1_OUT |
| 60 | | 60 | FTU3_CH2_OUT |
| 61 | | 61 | FTU3_CH3_OUT |
| 62 | | 62 | FTU3_CH4_OUT |
| 63 | | 63 | FTU3_CH5_OUT |
| 64 | | 64 | FTU3_CH6_OUT |
| 65 | | 65 | FTU3_CH7_OUT |
| 66 | | 66 | PTIMER0_ADCCH0_TRG |
| 67 | | 67 | PTIMER0_PULSE |
| 68 | | 68 | ADC0_EOSG[0] |
| 69 | | 69 | ADC0_EOSG[1] |
| 70 | | 70 | ADC0_EOSG[2] |
| 71 | | 71 | ADC0_EOSG[3] |
| 72 | | 72 | PTIMER1_ADCCH0_TRG |
| 73 | | 73 | PTIMER1_PULSE |
| 74 | | 74 | ADC1_EOSG[0] |
| 75 | | 75 | ADC1_EOSG[1] |
| 76 | | 76 | ADC1_EOSG[2] |
| 77 | | 77 | ADC1_EOSG[3] |
| 78 | | 78 | CMP0_OUT |
| 79 | | 79 | CMP1_OUT |
| 80 | | 80 | TPU_TRG0 |
| 81 | | 81 | TPU_TRG1 |
| 82 | | 82 | TPU_TRG2 |
| 83 | | 83 | TPU_TRG3 |
| 84 | | 84 | TPU_TRG4 |
| 85 | | 85 | TPU_TRG5 |
| 86 | | 86 | TPU_TRG6 |
| 87 | | 87 | TPU_TRG7 |
| 88 | | 88 | TPU_TRG8 |
| 89 | | 89 | TPU_TRG9 |
| 90 | | 90 | TPU_TRG10 |
| 91 | | 91 | TPU_TRG11 |
| 92 | | 92 | TPU_TRG12 |

Table 24. TRGSEL4 sources (continued)

| TRGSEL4 | | | |
|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** |
| 93 | | 93 | TPU_TRG13 |
| 94 | | 94 | TPU_TRG14 |
| 95 | | 95 | TPU_TRG15 |
| 96 | | 96 | TPU_TRG16 |
| 97 | | 97 | TPU_TRG17 |
| 98 | | 98 | TPU_TRG18 |
| 99 | | 99 | TPU_TRG19 |
| 100 | | 100 | TPU_TRG20 |
| 101 | | 101 | TPU_TRG21 |
| 102 | | 102 | TPU_TRG22 |
| 103 | | 103 | TPU_TRG23 |
| 104 | | 104 | TPU_TRG24 |
| 105 | | 105 | TPU_TRG25 |
| 106 | | 106 | TPU_TRG26 |
| 107 | | 107 | TPU_TRG27 |
| 108 | | 108 | TPU_TRG28 |
| 109 | | 109 | TPU_TRG29 |
| 110 | | 110 | TPU_TRG30 |
| 111 | | 111 | TPU_TRG31 |

#### *6.13.2.6 TRGSEL5*

Table 25. TRGSEL5 sources

| TRGSEL5 | | | |
|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** |
| 0 | TPU_INT_CH24 | 0 | VSS |
| 1 | TPU_INT_CH25 | 1 | VDD |
| 2 | TPU_INT_CH26 | 2 | SCM_SW_TRG4 |
| 3 | TPU_INT_CH27 | 3 | SCM_SW_TRG5 |
| 4 | TPU_INT_CH28 | 4 | SCM_SW_TRG6 |
| 5 | TPU_INT_CH29 | 5 | SCM_SW_TRG7 |
| 6 | TPU_INT_CH30 | 6 | TRGSEL_IN8 |
| 7 | TPU_INT_CH31 | 7 | TRGSEL_IN9 |
| 8 | FCPIT0_TRG_CH0 | 8 | TRGSEL_IN10 |
| 9 | FCPIT0_TRG_CH1 | 9 | TRGSEL_IN11 |
| 10 | FCPIT0_TRG_CH2 | 10 | TRGSEL_IN12 |
| 11 | FCPIT0_TRG_CH3 | 11 | TRGSEL_IN13 |
| 12 | AONTIMER0_CLK3 | 12 | ADC0_CMP |

**Table 25. TRGSEL5 sources (continued)**

| TRGSEL5 | | | |
|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** |
| 13 | | 13 | ADC1_CMP |
| 14 | | 14 | CMP0_OUT |
| 15 | | 15 | CMP1_OUT |
| 16 | | 16 | FTU0_RELOAD_TRG |
| 17 | | 17 | FTU0_MATCH_TRG |
| 18 | | 18 | FTU1_RELOAD_TRG |
| 19 | | 19 | FTU1_MATCH_TRG |
| 20 | | 20 | FTU2_RELOAD_TRG |
| 21 | | 21 | FTU2_MATCH_TRG |
| 22 | | 22 | FTU3_RELOAD_TRG |
| 23 | | 23 | FTU3_MATCH_TRG |
| 24 | | 24 | FTU4_RELOAD_TRG |
| 25 | | 25 | FTU4_MATCH_TRG |
| 26 | | 26 | FTU5_RELOAD_TRG |
| 27 | | 27 | FTU5_MATCH_TRG |
| 28 | | 28 | FTU6_RELOAD_TRG |
| 29 | | 29 | FTU6_MATCH_TRG |
| 30 | | 30 | FTU7_RELOAD_TRG |
| 31 | | 31 | FTU7_MATCH_TRG |
| 32 | | 32 | LU0_OUT0A |
| 33 | | 33 | LU0_OUT1A |
| 34 | | 34 | LU0_OUT2A |
| 35 | | 35 | LU0_OUT3A |
| 36 | | 36 | LU0_OUT0B |
| 37 | | 37 | LU0_OUT1B |
| 38 | | 38 | LU0_OUT2B |
| 39 | | 39 | LU0_OUT3B |
| 40 | | 40 | ADC0_EOSG[0] |
| 41 | | 41 | ADC0_EOSG[1] |
| 42 | | 42 | ADC0_EOSG[2] |
| 43 | | 43 | ADC0_EOSG[3] |
| 44 | | 44 | ADC1_EOSG[0] |
| 45 | | 45 | ADC1_EOSG[1] |
| 46 | | 46 | ADC1_EOSG[2] |
| 47 | | 47 | ADC1_EOSG[3] |
| 48 | | 48 | FTU0_CH0_OUT |
| 49 | | 49 | FTU0_CH1_OUT |
| 50 | | 50 | FTU0_CH2_OUT |
| 51 | | 51 | FTU0_CH3_OUT |

**Table 25. TRGSEL5 sources (continued)**

| TRGSEL5 | | | |
|---|---|---|---|
| **Output Trigger No.** | **Output Trigger** | **Input Trigger No.** | **Input Trigger** |
| 52 | | 52 | FTU0_CH4_OUT |
| 53 | | 53 | FTU0_CH5_OUT |
| 54 | | 54 | FTU0_CH6_OUT |
| 55 | | 55 | FTU0_CH7_OUT |
| 56 | | 56 | TRGSEL1_INPUT0 |
| 57 | | 57 | TRGSEL1_INPUT1 |
| 58 | | 58 | TRGSEL1_INPUT2 |
| 59 | | 59 | TRGSEL1_INPUT3 |
| 60 | | 60 | TRGSEL1_INPUT4 |
| 61 | | 61 | TRGSEL1_INPUT5 |
| 62 | | 62 | TRGSEL1_INPUT6 |
| 63 | | 63 | TRGSEL1_INPUT7 |

Flagchip

# Chapter 7 Peripheral Overview

## 7.1 Human-Machine Interface (HMI)

### 7.1.1 General-Purpose Input/Output (GPIO)

- Support the GPIO A/B/C/D/E control
- Support the input and output direction control
- Support operations such as the port input read, and output set/clear/toggle
- Support the multi master access permission control. Each pin can be assigned to one dedicated master

### 7.1.2 Port Control (PORT)

- Support IO control modules including PORT A/B/C/D/E
- Support the port emergency stop function
- Support the pin access control from different masters
- Support functions such as IO Driver Strength Enable (DSE)/ Slew Rate Enable (SRE))
- Support functions such as IO Pull Enable (PUE)/ Pull Select (PUS)
- Support the multi master access permission control. Each pin can be assigned to one dedicated master
- Support the separated pin interrupt for each core
- RESET/NMI share pads support the input filter function and pullup enable
- The JTAG_TDI and JTAG_TMS share pads will be pull up enable after reset; JTAG_TCLK share pad will be pull down enable after reset

***Note:*** The SOSC32K share pads PTE3/PTE14 are input-only pads. All functions on the two pads can be used for input only. The output function on the two pads is not supported.

## 7.2 Communication Interface

### 7.2.1 FLEX Controller Area Network (FLEXCAN)

- All FLEXCANs support CAN FD and PNET functions
- FLEXCAN supports 32 mailboxes and 10 enhanced FIFOs. The message buffer RAM supports 8 + 5 ECC structure
- Each mailbox supports 0 to 8, 16, 32, or 64 bytes data length
- Individual Rx Mask register per mailbox
- RAM not used by FLEXCAN can be used as the general-purpose RAM space
- Programmable loop-back mode supporting self-test operation
- Support listen-only mode
- The FLEXCAN functional clock, which is independent of the system clock, allows an application to adjust the FLEXCAN baud rate without impacting system frequency. Refer to PCC for detailed FLEXCAN clock settings
- Support the direct CAN connection function, allowing to connect multiple CANs without PHY

The multi-chip direct CAN connection is achieved through the below logic SCM CANINEN[15:0], which used to select which CAN TX will be connected through the TRGSEL_IN[15:0].

### 7.2.2 FC Serial Peripheral Interface (FCSPI)

- Support both master and slave modes. Support configurable clock polarity and clock phase
- Support to configure timing parameters in master mode, including delay between PCS and SCK, and the SCK frequency

- Support 8-word-deep FIFO with DMA capability
- Support 1-bit, 2-bit and 4-bit data transfer and multi-chip select in master mode (refer to pin out mux for detailed information). Not all functions are available in all cases
- The FCSPI functional clock, which is independent of the system clock, allows an application to adjust the FCSPI baud rate without impacting system frequency. Refer to PCC for detailed FCSPI clock settings
- Support the data match wake-up function

### 7.2.3 FC Inter-Integrated Circuit (FCIIC)

- Support standard and fast modes
- Support the 7-bit and 10-bit addressing
- Support the clock stretching function
- Support 8-word-deep FIFO with DMA capability
- The FCIIC functional clock, which is independent of the system clock, allows an application to adjust the FCIIC baud rate without impacting system frequency. Refer to PCC for detailed FCIIC clock settings

### 7.2.4 FC Universal Asynchronous Receiver Transmitter (FCUART)

- Support optional data length of 7-bit, 8-bit, 9-bit, and 10-bit
- Support the hardware parity function
- Support 1-bit or 2-bit stop bits
- Support the LIN break detect and break transmit
- Support hardware flow control (refer to pin out mux for details). Not all FCUARTs support this function
- Support 16-word-deep FIFO with DMA capability
- The FCUART functional clock, which is independent of the system clock, allows an application to adjust the FCUART baud rate without impacting system frequency. Refer to PCC for detailed FCUART clock settings

### 7.2.5 Micro Second Channel (MSC)

- Support high-speed synchronous serial transmission on a downstream channel
- Support a fractional clock divider for precise frequency control
- Support command, data and passive frame types
- Support low-speed asynchronous serial reception on an upstream channel
- Programmable upstream data frame length
- Parity error checker on an upstream channel

### 7.2.6 Single Edge Nibble Transmission (SENT)

- Conformance with SENT SAE J2716 2010 standard
- Up to 4 channels for each SENT
- Unified message read logic to transfer all messages received to the system memory via an interrupt or DMA
- All received messages are stored excluding the synchronization pulse (for fast messages) and identifiers (for slow messages)
- Separate buffers for fast/slow message
- Support all bit rates from 3 µs to 90 µs
- Support Short PWM Code (SPC) mode; the sensor only sends the data after receiving the SPC pulse
- Alternate CRC support for devices such as TLE4998

Flagchip

## 7.3 Timers

### 7.3.1 Flexible Timer Unit (FTU)

- FTU1 and FTU2 support 24-bit count
- Support a 16-bit counter
- Each FTU channel can be configured as input capture, output compare or Pulse Width Modulation (PWM) mode
- Support deadtime insertion
- Support match trigger generation
- Support 2 fault inputs. Sources of fault inputs are from TRGSEL2 outputs
- All FTUs support the enhanced input capture function

### 7.3.2 FC Programmable Interrupt Timer (FCPIT)

- Support up to 4 channels
- Support trigger generation and input trigger function
- The FCPIT functional clock is independent of the system clock

### 7.3.3 Always-on Timer (AONTIMER)

- 1-channel 16-bit counter
- Support working under standby mode if a clock source is available
- Configurable clock source and clock prescaler/filter

### 7.3.4 Timer Stamp (TSTMP)

- Support up to 4 TSTMP timers
- A free-running 56-bit counter
- TSTMP0 counter clock from the 1 MHz clock source; TSTMP1 counter clock from the bus clock source
- 4 × 32-bit timeout channels

### 7.3.5 Programmable Timer (PTIMER)

- Support both software and hardware triggers
- Output can be used to trigger ADC converting
- 16-bit delay counter
- Support back-to-back operation
- Support DMA function

### 7.3.6 Real-Time Clock (RTC)

- 32-bit seconds counter
- Support compensation function
- Support working under standby mode

### 7.3.7 Frequency Measurement (FREQM)

- Support prescaler on input clock sources
- Support up to 64 input clock sources

**Table 26. FREQM input sources**

| FREQM Clock Index | Clock Source | Comment |
|---|---|---|
| 63 | Reserved | |
| 62 | TPU functional clock | |
| 61 | MSC0 functional clock | |

Flagchip

**Table 26. FREQM input sources (continued)**

| FREQM Clock Index | Clock Source | Comment |
|---|---|---|
| 60 | Reserved | |
| 59 | FLEXCAN1 functional clock | |
| 58 | FLEXCAN0 functional clock | |
| 57 | SENT0 functional clock | |
| 56 | Reserved | |
| 55-48 | DIVH[7:0] | {1'b0, pll0, pll1, 1'b0, firc, sirc, fosc, 1'b0} |
| 47-40 | DIVM[7:0] | {1'b0, pll0, pll1, 1'b0, firc, sirc, fosc, 1'b0} |
| 39-32 | DIVL[7:0] | {1'b0, pll0, pll1, 1'b0, firc, sirc, fosc, 1'b0} |
| 31 | Reserved | |
| 30 | Reserved | |
| 29 | ADC1 functional clock | |
| 28 | Reserved | |
| 27 | ADC0 functional clock | |
| 26 | FTU3 functional clock | |
| 25 | FTU2 functional clock | |
| 24 | Reserved | |
| 23 | FTU1 functional clock | |
| 22 | FTU0 functional clock | |
| 21 | TRGSEL1_TRGSEL_CLK | |
| 20 | Reserved | |
| 19 | FCUART2 functional clock | |
| 18 | FCUART1 functional clock | |
| 17 | FCUART0 functional clock | |
| 16 | Reserved | |
| 15 | FCSPI2 functional clock | |
| 14 | FCSPI1 functional clock | |
| 13 | FCSPI0 functional clock | |
| 12 | Reserved | |
| 11 | Reserved | |
| 10 | FCPIT0 functional clock | |
| 9 | Reserved | |
| 8 | Reserved | |
| 7 | SIRC 128K_CLK | |
| 6 | AON_CLK | |
| 5 | RTC_CLK | |
| 4 | SCG_CLKOUT | |
| 3 | PLL0_FEEDBACK | |
| 2 | PLL1_FEEDBACK | |
| 1 | Slow Clock | |
| 0 | Reserved | |

## 7.4 Analog

### 7.4.1 Analog-to-Digital Converter (ADC)

- Support 12-bit, 10-bit, and 8-bit resolution
- Support single conversion or continuous conversion modes
- Support 32 external analog inputs and 8 internal channels
- Support up to 30 MHz sampling clock
- Support single-ended channels only
- Support compare function
- Support hardware/software trigger function; support 4 hardware trigger function
- Support hardware trigger with configured convert group
- Support hardware trigger error detection
- The ADC functional clock, which is independent of the system clock, allows an application to adjust the ADC sample rate without impacting system frequency. Refer to PCC for detailed ADC clock settings
- The sampling time for each ADC channel is configurable

### 7.4.2 Comparator (CMP)

- Support internal 8-bit DAC
- Support up to 8 channels. Refer to pin mux table for detailed mapping
- Programmable hysteresis control
- Digital filter on CMP hard output
- CMP0 and CMP1 support 1.2 V VREF or VDDA as the DAC reference
- Support the Channel Scan function
- Support the auto mask CMP out if changing the CMP setting with CMP enable

### 7.4.3 Internal Analog Channels Connection

**Table 27. ADC0 Internal analog channels connection**

| Signal Name | ADC Channels |
|---|---|
| VBG Buffer | 32 |
| V25 | 33 |
| V11 | 34 |
| N/A | 35 |
| CMP0 DAC | 36 |
| tempsensor_ts_out | 37 |

**Table 28. ADC1 Internal analog channels connection**

| Signal Name | ADC Channels |
|---|---|
| V11_PD0 | 32 |
| VREFH | 33 |
| VDDA | 34 |
| V15 | 35 |
| CMP DAC0 | 36 |
| N/A | 37 |

Flagchip

# Chapter 8 Embedded Memory

This section summarizes the FC7240F2M embedded memory configurations. It includes details on flash, RAM, system register file and ROM implementation.

## 8.1 Flash Memory

The information related to the flash memory of FC7240F2M is listed below:

| | PFlash | DFlash | OTA | Sector Size | | Max. Read Frequency | Min. Program Size | | Bit Width | Read Line Buffer |
|---|---|---|---|---|---|---|---|---|---|---|
| **FC7240F2M** | 2 × 1 MB | 128 KB | Yes | PFlash (64-bit) | 2 KB | 33.33 MHz | PFlash (64-bit) | 8 bytes at a time | 64-bit (parallel) | 4 × 32 bytes |
| | | | | DFlash | | | DFlash | | | |
| | | | | PFlash (128-bit) | 4 KB | | PFlash (128-bit) | 16 bytes at a time | 128-bit (non-parallel) | |

### 8.1.1 Flash Swap Function

The OTA hardware control register supports up to 2 × 1MB flash OTA. Refer to *Chapter FMC* for detailed function description. Flash Deep Power Down (DPD) is supported during the Stop mode.

The FC7240F2M can optionally support OTA (64-bit) or non-OTA (128-bit) read. For 64-bit/128-bit settings, see descriptions of SEC_BCS[PF_128BIT_Mode] in *Chapter SEC.* Refer to *Section 12.1 NVR Control (Debug etc.)* for detailed OTA information.

### 8.1.2 Flash Configuration Mapping

The flash has an extra sector NVR. On the FC7240F2M, the NVR is 40 KB. Refer to Flash configuration for detailed information/usage. The NVR sector can be erased/programed by users. If the secure boot is enable, the NVR sector will be not able to be read by CPU, and only HSM is allowed to be read.

For the FC7240F2M, NVR has three segments:

- NVR0, 16 KB: 0x0440_0000 to 0x0440_3FFF
- NVR1, 16 KB: 0x0440_8000 to 0x0440_BFFF
- NVR2, 8 KB: 0x0441_0000 to 0x0441_1FFF

### 8.1.3 Flash ECC

On the FC7240F2M, PFlash uses the 64 + 16bit structure, and DFlash uses the single 64 + 16bit structure. The address ECC is included in the ECC check bits. To overcome the un-program flash issue, extra bits are used to indicate if the flash is in the program or un-program state. The ECC output is re-encoded and compared with ECC bits to make sure the Flash ECC logic function is correct.

### 8.1.4 Flash Memory Control

The FC7240F2M Flash Memory Control support the 128-bit ECC for mini-linebuffer (for non-OTA cases) or 64-bit ECC for mini-linebuffer (for OTA cases). The address ECC is included in ECC check bits. For data flash, it has no mini-linebuffer, but only supports prefetch.

## 8.2 SRAM Memory

The FC7240F2M contains a 96 KB SRAM, and a 32 KB ITCM and 128 KB DTCM for CM7. SRAM_0 is 2 × 32KB, from 0x2100_0000 to 0x2100_FFFF, can be optionally retained during Standby mode.

***Note:*** If booting from ROM, the first 3 KB space (0x2000_0000 - 0x2000_0BFF) of the DTCM0 will be occupied by BootROM. After each reset, the 3 KB space must be initialized before it can be used.

The ITCM/DTCM/SRAM ECC support the Address and Data ECC.

## 8.3 ROM

The FC7240F2M contains a 128 KB ROM, comprising the CM7 core self-test API, flash API, and secure boot code. The ROM ECC supports both data and address protection.

Flagchip

# Chapter 9 Clock Overview

This chapter summarizes the FC7240F2M clock architecture. It includes a high-level clocking diagram and internal clocking requirements. It has 5 CMUs, which use SIRC to monitor the FIRC/SOSC/FOSC and use FIRC to monitor the SIRC. Also, the CMU4 uses the FOSC to monitor the slow clock.

## 9.1 System Clock Generator (SCG)

The SCG provides the system clock control and system clock generation. It also provides the loss of clock detection for the PLL0, PLL1, FOSC, and bus clock, and loss of lock detection for the PLL0 and PLL1.

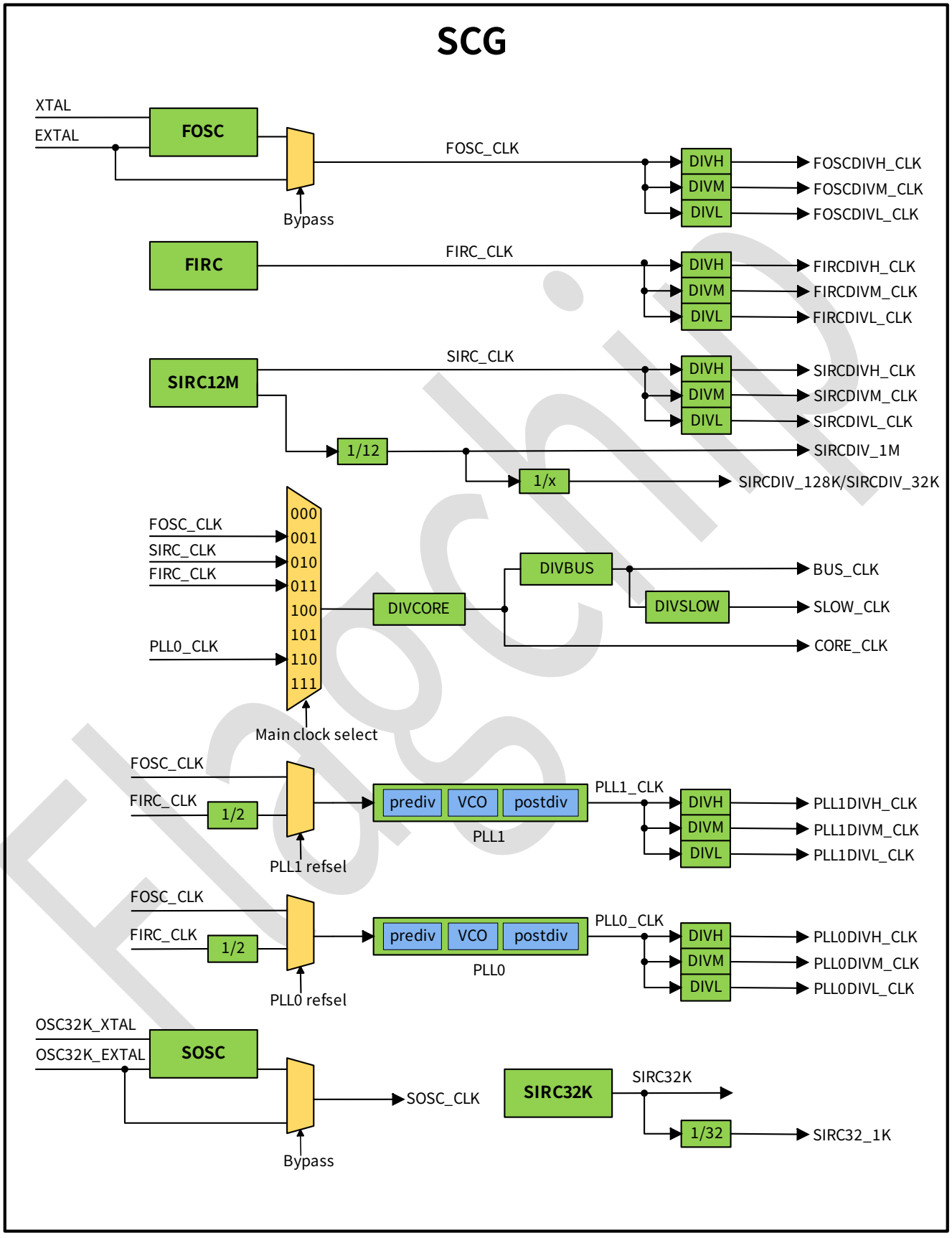
The figure below shows a top-level clocking diagram of the superset device. For detailed clocking diagram of the FC7240, refer to the attached *FC7240 Clocking Diagram.pdf.*

The system clock root source includes the FIRC96M/PLL0/FOSC16–48M or the external input clock. For low-power applications (non-safety cases), the SIRC12M can be selected as the system clock too. The SIRC12M/ SOSC32k/SIRC32k and PLL1 can be used as peripherals' (such as timer and communication) clocks.

FIRC/FOSC/PLL0/PLL1/SIRC has 3 functional clock divider outputs, which can be used as clock sources for other peripherals. The functional clocks can be configured through the corresponding DIV register. The three divider outputs are named DIVH, DIVM, and DIVL. For the FC7240F2M, the clock output after DIVH should be not above 160 MHz, DIVM should be not above 160 MHz, and DIVL should be not above 80 MHz. The maximum frequency of PLL0/PLL1 is 240 MHz/320 MHz respectively.

Flagchip

**Figure 12. FC7240F2M top-level clocking diagram**




The PLL0 and PLL1 VCO clock should fall in the range of 240 MHz to 640 MHz (and PLL clock output is VCO/2). The PLL reference clock is 16 – 48 MHz. And the PLL feedback clock should fall between 2 – 4 MHz.

SIRCDIV_128K, SIRCDIV_32K and SIRCDIV_1M are divided from SIRC12M. SIRC32_1K is divided from SIRC32K.

## 9.2 Peripheral Clocks

Peripherals' clocks are controlled by the PCC. The PCC selects the peripherals' functional clocks from 8 input clock sources. The PCC of some IPs, including FLEXCANx, AONTIMER, ADCx, SENT, and MSC, contains a clock divider.

### 9.2.1 PCC Assignments

The FC7240F2M contains one PCC block. The below table shows the detailed peripheral PCC assignments. The PCC comprises options including CGC, Clock Mux, and Divide, and it contains different settings for different peripherals.

- The MUX_DIVH means the PCC clock MUX uses the FIRC/PLL1/PLL0/FOSC/SIRC DIVH clock output as the source.
- The MUX_DIVM means the PCC clock MUX uses the FIRC/PLL1/PLL0/FOSC/SIRC DIVM clock output as the source.
- The MUX_DIVL means the PCC clock MUX uses the FIRC/PLL1/PLL0/FOSC/SIRC DIVL clock output as the source.
- The MUX_DIVx_PIN means MUX input 0 is from the pin input.

#### *9.2.1.1 PCC0 Part0 Specification*

The below table lists the peripheral clock assignments of the PCC0 part0 slot.

**Table 29. PCC0 part0 specification**

| | PCC Index | Interface Clock | Divide Available | CGC Available | CGC Bit Reset | Clock MUX Type | DWP/SWR Available | DWP Reset |
|---|---|---|---|---|---|---|---|---|
| Reserved | 0 | - | - | - | - | - | - | |
| Reserved | 1 | - | - | - | - | - | - | |
| Reserved | 2 | - | - | - | - | - | - | |
| Reserved | 3 | - | - | - | - | - | - | |
| Reserved | 4 | - | - | - | - | - | - | |
| Reserved | 5 | - | - | - | - | - | - | |
| Reserved | 6 | - | - | - | - | - | - | |
| Reserved | 7 | - | - | - | - | - | - | |
| DMA0 | 8 | Bus | - | Yes | - | - | Yes | |
| Reserved | 9 | - | - | - | - | - | - | |
| DMAMUX0 | 10 | Bus | - | Yes | - | - | Yes | |
| Reserved | 11 | - | - | - | - | - | - | |
| Reserved | 12 | - | - | - | - | - | - | |
| Reserved | 13 | - | - | - | - | - | - | |
| Reserved | 14 | - | - | - | - | - | - | |
| FCSMU | 15 | Bus | - | - | - | - | - | |
| Reserved | 16 | - | - | - | - | - | - | |
| Reserved | 17 | - | - | - | - | - | - | |
| Reserved | 18 | - | - | - | - | - | - | |
| ROMC | 19 | Bus | - | Yes | 1 | - | Yes | |
| SEC | 20 | Slow | - | - | - | - | - | |
| Reserved | 21 | - | - | - | - | - | - | |
| Reserved | 22 | - | - | - | - | - | - | |
| Reserved | 23 | - | - | - | - | - | - | |

**Table 30. PCC0 part0 specification (continued)**

| | PCC Index | Interface Clock | Divide Available | CGC Available | CGC Bit Reset | Clock MUX Type | DWP/SWR Available | DWP Reset |
|---|---|---|---|---|---|---|---|---|
| ERM | 24 | Bus | - | Yes | 1 | - | Yes | |
| EIM | 25 | Slow | - | Yes | - | - | Yes | |
| INTM0 | 26 | Slow | - | Yes | - | - | Yes | CPU0 |
| ISM0 | 27 | Bus | - | Yes | - | - | Yes | |
| Reserved | 28 | - | - | - | - | - | - | |
| Reserved | 29 | | - | - | - | - | - | |
| Reserved | 30 | | - | - | - | - | - | |
| Reserved | 31 | | - | - | - | - | - | |
| FC | 32 | Bus | - | - | - | - | - | |
| FCA | 33 | Bus | - | - | - | - | - | |
| WDOG0 | 34 | Slow | - | - | - | - | Yes | |
| SCG | 35 | Slow | - | - | - | - | - | |
| PCC0 | 36 | Slow | - | - | - | - | - | |
| Reserved | 37 | - | - | - | - | - | - | |
| TRGSEL0 | 38 | Slow | - | Yes | - | - | Yes | |
| TRGSEL1 | 39 | Slow | - | Yes | - | - | Yes | |
| TRGSEL2 | 40 | Slow | - | Yes | - | - | Yes | |
| TRGSEL3 | 41 | Slow | - | Yes | - | - | Yes | |
| CRC0 | 42 | Bus | - | Yes | - | - | Yes | |
| CORDIC0 | 43 | Bus | - | Yes | - | - | Yes | |
| TSTMP0 | 44 | Bus | - | Yes | - | - | Yes | |
| TSTMP1 | 45 | Bus | - | Yes | - | - | Yes | |
| FCPIT0 | 46 | Bus | - | Yes | - | MUX_DIVH | Yes | |
| AONTIMER0 | 47 | Slow | Yes | Yes | - | MUX_DIVL | Yes | |
| RTC | 48 | Slow | - | - | - | - | Yes | |
| CMU0 | 49 | Slow | - | Yes | - | | Yes | CPU0 |
| CMU1 | 50 | Slow | - | Yes | - | | Yes | CPU0 |
| CMU2 | 51 | Slow | - | Yes | - | | Yes | CPU0 |
| CMU3 | 52 | Slow | - | Yes | - | | Yes | CPU0 |
| CMU4 | 53 | Slow | - | Yes | - | | Yes | CPU0 |
| Reserved | 54 | - | - | - | - | - | - | |
| PTIMER0 | 55 | Bus | - | Yes | - | - | Yes | |
| PTIMER1 | 56 | Bus | - | Yes | - | - | Yes | |
| Reserved | 57 | - | - | - | - | - | - | |
| Reserved | 58 | - | - | - | - | - | - | |
| ADC0 | 59 | Bus | Yes | Yes | - | MUX_DIVH | Yes | |
| ADC1 | 60 | Bus | Yes | Yes | - | MUX_DIVH | Yes | |
| Reserved | 61 | - | - | - | - | - | - | |
| Reserved | 62 | - | - | - | - | - | - | |

**Table 30. PCC0 part0 specification (continued)**

| | PCC Index | Interface Clock | Divide Available | CGC Available | CGC Bit Reset | Clock MUX Type | DWP/SWR Available | DWP Reset |
|---|---|---|---|---|---|---|---|---|
| WKU | 63 | Slow | - | Yes | - | | Yes | |
| CMP0 | 64 | Slow | - | Yes | - | - | Yes | |
| CMP1 | 65 | Slow | - | Yes | - | - | Yes | |
| Reserved | 66 | - | - | - | - | - | - | |
| TMU | 67 | Slow | - | Yes | - | MUX_DIVL | Yes | |
| PMC | 68 | Slow | - | - | - | | - | |
| SMC | 69 | Slow | - | - | - | | - | |
| RGM | 70 | Slow | - | - | - | | - | CPU0 |
| Port A | 71 | Slow | - | - | - | | - | |
| Port B | 72 | Slow | - | - | - | | - | |
| Port C | 73 | Slow | - | - | - | | - | |
| Port D | 74 | Slow | - | - | - | | - | |
| Port E | 75 | Slow | - | - | - | | - | |
| Reserved | 76 | - | - | - | - | | - | |
| Reserved | 77 | - | - | - | - | | - | |
| Reserved | 78 | - | - | - | - | | - | |
| Reserved | 79 | - | - | - | - | | - | |
| Reserved | 80 | - | - | - | - | | - | |
| Reserved | 81 | - | - | - | - | | - | |
| Reserved | 82 | - | - | - | - | | - | |
| Reserved | 83 | - | - | - | - | | - | |
| SENT0 | 84 | Bus | Yes | Yes | - | MUX_DIVL | Yes | |
| Reserved | 85 | - | - | - | - | - | - | |
| Reserved | 86 | - | - | - | - | - | - | |
| Reserved | 87 | - | - | - | - | - | - | |
| MB | 88 | Bus | - | - | - | - | Yes | |
| Reserved | 89 | - | - | - | - | - | - | |
| Reserved | 90 | - | - | - | - | - | - | |
| Reserved | 91 | - | - | - | - | - | - | |
| FTU0 | 92 | Bus | - | Yes | - | MUX_DIVH_PIN | Yes | |
| FTU1 | 93 | Bus | - | Yes | - | MUX_DIVH_PIN | Yes | |
| FTU2 | 94 | Bus | - | Yes | - | MUX_DIVH_PIN | Yes | |
| FTU3 | 95 | Bus | - | Yes | - | MUX_DIVH_PIN | Yes | |
| Reserved | 96 | - | - | - | - | - | - | |
| Reserved | 97 | - | - | - | - | - | - | |
| FCSPI0 | 98 | Bus | - | Yes | - | MUX_DIVM | Yes | |
| FCSPI1 | 99 | Bus | - | Yes | - | MUX_DIVM | Yes | |
| FCSPI2 | 100 | Bus | - | Yes | - | MUX_DIVM | Yes | |
| Reserved | 101 | Bus | - | Yes | - | MUX_DIVM | Yes | |

**Table 30. PCC0 part0 specification (continued)**

| | PCC Index | Interface Clock | Divide Available | CGC Available | CGC Bit Reset | Clock MUX Type | DWP/SWR Available | DWP Reset |
|---|---|---|---|---|---|---|---|---|
| FCIIC0 | 102 | Bus | - | Yes | - | MUX_DIVM | Yes | |
| Reserved | 103 | - | - | - | - | - | - | |
| FCUART0 | 104 | Bus | - | Yes | - | MUX_DIVM | Yes | |
| FCUART1 | 105 | Bus | - | Yes | - | MUX_DIVM | Yes | |
| FCUART2 | 106 | Bus | - | Yes | - | MUX_DIVM | Yes | |
| FCUART3 | 107 | Bus | - | Yes | - | MUX_DIVM | Yes | |
| Reserved | 108 | - | - | - | - | - | - | |
| Reserved | 109 | - | - | - | - | - | - | |
| Reserved | 110 | - | - | - | - | - | - | |
| Reserved | 111 | - | - | - | - | - | - | |
| LU0 | 112 | Slow | - | Yes | - | | Yes | |
| Reserved | 113 | Slow | - | Yes | - | | Yes | |
| SCM | 114 | Bus | - | - | - | - | - | |
| Reserved | 115 | - | - | - | - | - | - | |
| Reserved | 116 | - | - | - | - | - | - | |
| Reserved | 117 | - | - | - | - | - | - | |
| Reserved | 118 | - | - | - | - | - | - | |
| Reserved | 119 | - | - | - | - | - | - | |
| FREQM | 120 | Bus | - | Yes | - | - | Yes | |
| CSC0 | 121 | Bus | - | - | - | - | - | |
| Reserved | 122 | - | - | - | - | - | - | |
| Reserved | 123 | - | - | - | - | - | - | |
| Reserved | 124 | - | - | - | - | - | - | |
| Reserved | 125 | - | - | - | - | - | - | |
| Reserved | 126 | - | - | - | - | - | - | |
| STCU | 127 | Slow | - | Yes | - | - | Yes | |
| FLEXCAN0 | 128 | Bus | Yes | Yes | - | MUX_DIVH | Yes | |
| | 129 | - | - | - | - | - | - | |
| | 130 | - | - | - | - | - | - | |
| | 131 | - | - | - | - | - | - | |
| FLEXCAN1 | 132 | Bus | Yes | Yes | - | MUX_DIVH | Yes | |
| | 133 | - | - | - | - | - | - | |
| | 134 | - | - | - | - | - | - | |
| | 135 | - | - | - | - | - | - | |
| Reserved | 136 | - | - | - | - | - | - | |
| | 137 | - | - | - | - | - | - | |
| | 138 | - | - | - | - | - | - | |
| | 139 | - | - | - | - | - | - | |
| Reserved | 140 | - | - | - | - | - | - | |

**Table 30. PCC0 part0 specification (continued)**

| | PCC Index | Interface Clock | Divide Available | CGC Available | CGC Bit Reset | Clock MUX Type | DWP/SWR Available | DWP Reset |
|---|---|---|---|---|---|---|---|---|
| | 141 | - | - | - | - | - | - | |
| | 142 | - | - | - | - | - | - | |
| | 143 | - | - | - | - | - | - | |
| Reserved | 144 | - | - | - | - | - | - | |
| | 145 | - | - | - | - | - | - | |
| | 146 | - | - | - | - | - | - | |
| | 147 | - | - | - | - | - | - | |
| Reserved | 148 | - | - | - | - | - | - | |
| | 149 | - | - | - | - | - | - | |
| | 150 | - | - | - | - | - | - | |
| | 151 | - | - | - | - | - | - | |
| Reserved | 152 | - | - | - | - | - | - | |
| | 153 | - | - | - | - | - | - | |
| | 154 | - | - | - | - | - | - | |
| | 155 | - | - | - | - | - | - | |
| Reserved | 156 | - | - | - | - | - | - | |
| | 157 | - | - | - | - | - | - | |
| | 158 | - | - | - | - | - | - | |
| | 159 | - | - | - | - | - | - | |

#### *9.2.1.2 PCC0 Part1 Specification*

The below table lists the peripheral clock assignments of the PCC0 part 1 slot.

**Table 30. PCC0 part1 specification**

| | PCC Index | Interface Clock | Divide Available | CGC Available | CGC Bit Reset | Clock MUX Type | DWP/SWR Available | DWP Reset |
|---|---|---|---|---|---|---|---|---|
| Reserved | 160 | - | - | - | - | - | - | |
| Reserved | 161 | - | - | - | - | - | - | |
| Reserved | 162 | - | - | - | - | - | - | |
| Reserved | 163 | - | - | - | - | - | - | |
| Reserved | 164 | - | - | - | - | - | - | |
| Reserved | 165 | - | - | - | - | - | - | |
| Reserved | 166 | - | - | - | - | - | - | |
| Reserved | 167 | - | - | - | - | - | - | |
| Reserved | 168 | - | - | - | - | - | - | |
| Reserved | 169 | - | - | - | - | - | - | |
| Reserved | 170 | - | - | - | - | - | - | |
| Reserved | 171 | - | - | - | - | - | - | |
| Reserved | 172 | - | - | - | - | - | - | |

**Table 31. PCC0 part1 specification (continued)**

| | PCC Index | Interface Clock | Divide Available | CGC Available | CGC Bit Reset | Clock MUX Type | DWP/SWR Available | DWP Reset |
|---|---|---|---|---|---|---|---|---|
| MAM0 | 173 | Bus | - | - | - | - | - | |
| Reserved | 174 | - | - | - | - | - | - | |
| Reserved | 175 | - | - | - | - | - | - | |
| Reserved | 176 | - | - | - | - | - | - | |
| Reserved | 177 | - | - | - | - | - | - | |
| Reserved | 178 | - | - | - | - | - | - | |
| Reserved | 179 | - | - | - | - | - | - | |
| Reserved | 180 | - | - | - | - | - | - | |
| Reserved | 181 | - | - | - | - | - | - | - |
| Reserved | 182 | - | - | - | - | - | - | - |
| Reserved | 183 | - | - | - | - | - | - | |
| Reserved | 184 | - | - | - | - | - | - | |
| Reserved | 185 | - | - | - | - | - | - | |
| Reserved | 186 | - | - | - | - | - | - | |
| Reserved | 187 | - | - | - | - | - | - | |
| Reserved | 188 | - | - | - | - | - | - | |
| Reserved | 189 | - | - | - | - | - | - | |
| Reserved | 190 | - | - | - | - | - | - | |
| Reserved | 191 | - | - | - | - | - | - | |
| Reserved | 192 | - | - | - | - | - | - | |
| Reserved | 193 | - | - | - | - | - | - | |
| Reserved | 194 | - | - | - | - | - | - | |
| Reserved | 195 | - | - | - | - | - | - | |
| Reserved | 196 | - | - | - | - | - | - | |
| Reserved | 197 | - | - | - | - | - | - | |
| Reserved | 198 | - | - | - | - | - | - | |
| Reserved | 199 | - | - | - | - | - | - | |
| Reserved | 200 | - | - | - | - | - | - | |
| Reserved | 201 | - | - | - | - | - | - | |
| Reserved | 202 | - | - | - | - | - | - | |
| Reserved | 203 | - | - | - | - | - | - | |
| Reserved | 204 | - | - | - | - | - | - | |
| Reserved | 205 | - | - | - | - | - | - | |
| Reserved | 206 | - | - | - | - | - | - | |
| Reserved | 207 | - | - | - | - | - | - | |
| Reserved | 208 | - | - | - | - | - | - | |
| Reserved | 209 | - | - | - | - | - | - | |
| Reserved | 210 | - | - | - | - | - | - | |
| WDOG1 | 211 | Slow | - | - | - | - | Yes | |

**Table 31. PCC0 part1 specification (continued)**

| | PCC Index | Interface Clock | Divide Available | CGC Available | CGC Bit Reset | Clock MUX Type | DWP/SWR Available | DWP Reset |
|---|---|---|---|---|---|---|---|---|
| Reserved | 212 | - | - | - | - | - | - | |
| Reserved | 213 | - | - | - | - | - | - | |
| Reserved | 214 | - | - | - | - | - | - | |
| Reserved | 215 | - | - | - | - | - | - | |
| Reserved | 216 | - | - | - | - | - | - | |
| Reserved | 217 | - | - | - | - | - | - | |
| Reserved | 218 | - | - | - | - | - | - | |
| TRGSEL4 | 219 | Slow | - | Yes | - | - | Yes | |
| TRGSEL5 | 220 | Slow | - | Yes | - | - | Yes | |
| Reserved | 221 | - | - | - | - | - | - | |
| Reserved | 222 | - | - | - | - | - | - | |
| FCSPI3 | 223 | Bus | | Yes | | MUX_DIVM | Yes | |
| FCSPI4 | 224 | Bus | | Yes | | MUX_DIVM | Yes | |
| FCSPI5 | 225 | Bus | | Yes | | MUX_DIVM | Yes | |
| Reserved | 226 | - | - | - | - | - | - | |
| Reserved | 227 | - | - | - | - | - | - | |
| Reserved | 228 | - | - | - | - | - | - | |
| TPUH0 | 229 | BUS | | Yes | | | Yes | |
| TPUE0 | 230 | | | | | | | |
| Reserved | 231 | - | - | - | - | - | - | |
| Reserved | 232 | - | - | - | - | - | - | |
| Reserved | 233 | - | - | - | - | - | - | |
| Reserved | 234 | - | - | - | - | - | - | |
| Reserved | 235 | - | - | - | - | - | - | |
| Reserved | 236 | - | - | - | - | - | - | |
| Reserved | 237 | - | - | - | - | - | - | |
| Reserved | 238 | - | - | - | - | - | - | |
| Reserved | 239 | - | - | - | - | - | - | |
| Reserved | 240 | - | - | - | - | - | - | |
| Reserved | 241 | - | - | - | - | - | - | |
| Reserved | 242 | - | - | - | - | - | - | |
| Reserved | 243 | - | - | - | - | - | - | |
| Reserved | 244 | - | - | - | - | - | - | |
| Reserved | 245 | - | - | - | - | - | - | |
| Reserved | 246 | - | - | - | - | - | - | |
| Reserved | 247 | - | - | - | - | - | - | |
| Reserved | 248 | - | - | - | - | - | - | |
| Reserved | 249 | - | - | - | - | - | - | |
| Reserved | 250 | - | - | - | - | - | - | |

**Table 31. PCC0 part1 specification (continued)**

| | PCC Index | Interface Clock | Divide Available | CGC Available | CGC Bit Reset | Clock MUX Type | DWP/SWR Available | DWP Reset |
|---|---|---|---|---|---|---|---|---|
| Reserved | 251 | - | - | - | - | - | - | |
| Reserved | 252 | - | - | - | - | - | - | |
| Reserved | 253 | - | - | - | - | - | - | |
| Reserved | 254 | - | - | - | - | - | - | |
| FTU4 | 255 | Bus | - | Yes | - | MUX_DIVH_PIN | Yes | |
| FTU5 | 256 | Bus | - | Yes | - | MUX_DIVH_PIN | Yes | |
| FTU6 | 257 | Bus | - | Yes | - | MUX_DIVH_PIN | Yes | |
| FTU7 | 258 | Bus | - | Yes | - | MUX_DIVH_PIN | Yes | |
| Reserved | 259 | - | - | - | - | - | - | |
| Reserved | 260 | - | - | - | - | - | - | |
| Reserved | 261 | - | - | - | - | - | - | |
| Reserved | 262 | - | - | - | - | - | - | |
| FCIIC1 | 263 | Bus | - | Yes | - | MUX_DIVM | Yes | |
| FCUART4 | 264 | Bus | - | Yes | - | MUX_DIVM | Yes | |
| FCUART5 | 265 | Bus | - | Yes | - | MUX_DIVM | Yes | |
| FCUART6 | 266 | Bus | - | Yes | - | MUX_DIVM | Yes | |
| FCUART7 | 267 | Bus | - | Yes | - | MUX_DIVM | Yes | |
| Reserved | 268 | - | - | - | - | - | - | |
| Reserved | 269 | - | - | - | - | - | - | |
| Reserved | 270 | - | - | - | - | - | - | |
| Reserved | 271 | - | - | - | - | - | - | |
| Reserved | 272 | - | - | - | - | - | - | |
| Reserved | 273 | - | - | - | - | - | - | |
| Reserved | 274 | - | - | - | - | - | - | |
| Reserved | 275 | - | - | - | - | - | - | |
| MSC0 | 276 | Bus | Yes | Yes | - | MUX_DIVM | Yes | |
| Reserved | 277 | - | - | - | - | - | - | |
| Reserved | 278 | - | - | - | - | - | - | |
| Reserved | 279 | - | - | - | - | - | - | |
| Reserved | 280 | - | - | - | - | - | - | |
| Reserved | 281 | - | - | - | - | - | - | |
| Reserved | 282 | - | - | - | - | - | - | |
| Reserved | 283 | - | - | - | - | - | - | |
| Reserved | 284 | - | - | - | - | - | - | |
| Reserved | 285 | - | - | - | - | - | - | |
| Reserved | 286 | - | - | - | - | - | - | |
| Reserved | 287 | - | - | - | - | - | - | |
| FLEXCAN2 | 288 | Bus | Yes | Yes | - | MUX_DIVH | Yes | |

**Table 31. PCC0 part1 specification (continued)**

| | PCC Index | Interface Clock | Divide Available | CGC Available | CGC Bit Reset | Clock MUX Type | DWP/SWR Available | DWP Reset |
|---|---|---|---|---|---|---|---|---|
| | 289 | - | - | - | - | - | - | |
| | 290 | - | - | - | - | - | - | |
| | 291 | - | - | - | - | - | - | |
| FLEXCAN3 | 292 | Bus | Yes | Yes | - | MUX_DIVH | Yes | |
| | 293 | - | - | - | - | - | - | |
| | 294 | - | - | - | - | - | - | |
| | 295 | - | - | - | - | - | - | |
| Reserved | 296 | - | - | - | - | - | - | |
| | 297 | - | - | - | - | - | - | |
| | 298 | - | - | - | - | - | - | |
| | 299 | - | - | - | - | - | - | |
| Reserved | 300 | - | - | - | - | - | - | |
| | 301 | - | - | - | - | - | - | |
| | 302 | - | - | - | - | - | - | |
| | 303 | - | - | - | - | - | - | |
| Reserved | 304 | - | - | - | - | - | - | |
| | 305 | - | - | - | - | - | - | |
| | 306 | - | - | - | - | - | - | |
| | 307 | - | - | - | - | - | - | |
| Reserved | 308 | - | - | - | - | - | - | |
| | 309 | - | - | - | - | - | - | |
| | 310 | - | - | - | - | - | - | |
| | 311 | - | - | - | - | - | - | |
| Reserved | 312 | - | - | - | - | - | - | |
| | 313 | - | - | - | - | - | - | |
| | 314 | - | - | - | - | - | - | |
| | 315 | - | - | - | - | - | - | |
| HSM0 | 316 | Bus | | Yes | | | Yes | |
| | 317 | - | - | - | - | - | - | |
| | 318 | - | - | - | - | - | - | |
| | 319 | - | - | - | - | - | - | |

### 9.2.2 Clock Structure

The below shows some peripherals' clock structure.

#### *9.2.2.1 Miscellaneous Clock*

The port pin input filter uses AON32K_CLK as a pin input glitch filter during low power mode.

The stop acknowledgement timeout uses the AON32K_CLK as the clock source.

The reset pin filter uses AON32K_CLK as the filter during low power mode.

The CMU period measure trigger is from SIRC32_1K which is divided from SIRC32K.

#### 9.2.2.2 TSTMP Clock

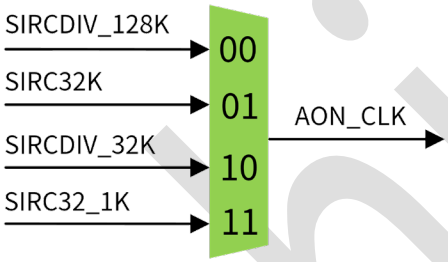
The TSTMP0 functional clock is from SIRCDIV_1M divided from SIRC12M, and is always enabled during run mode and configurable to keep enabled during standby mode.

The TSTMP1-3 functional clock is from the bus clock.

#### 9.2.2.3 WDG AON_CLK Clock

The AON_CLK clock is from the SIRC12M divider output or SIRC32K. The clock selection signals are from the CSC0. This clock is also used as the trigger source of the internal CRC self-check hardware for FCSMU etc.

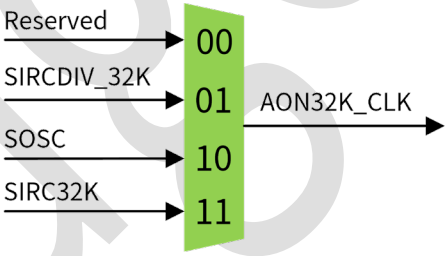
**Figure 13. AON_CLK clock**




#### 9.2.2.4 AON32K_CLK Clock

The AON32K_CLK clock is from SIRC32K, SOSC, or SIRCDIV_32K (SIRC12M divider output). The clock control signals are from the CSC0. The RESET and STOP acknowledgement timeouts use this clock source. The AON32K_CLK clock selects SIRCDIV_32K as the source by default.

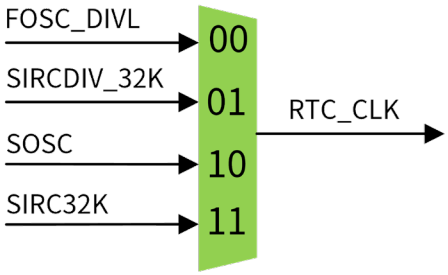
**Figure 14. AON32K clock**




#### 9.2.2.5 RTC Clock

The RTC clock is from FOSC DIVL, SIRC32K, SOSC, or SIRCDIV_32K (SIRC12M divider output). The clock control signals are from the CSC0. The RTC clock selects SIRCDIV_32K as the source by default.

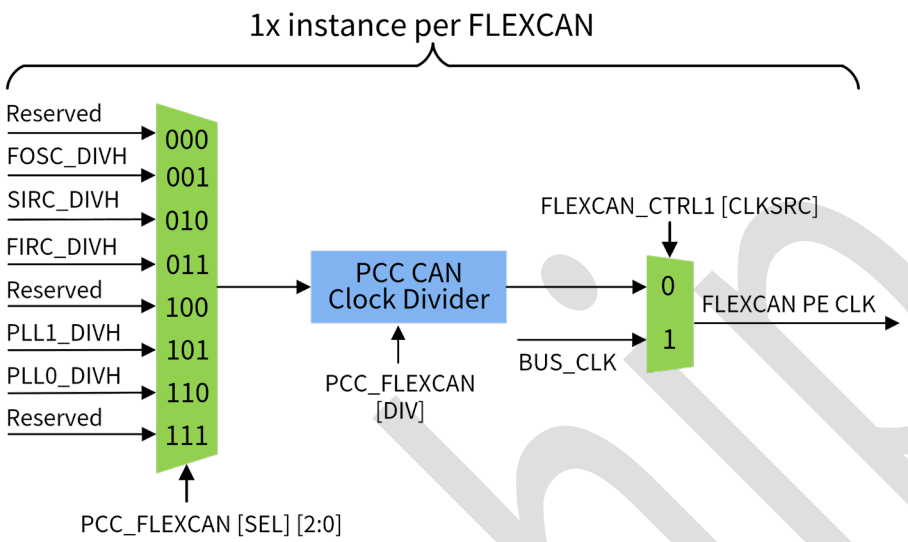
**Figure 15. RTC 32K clock**



#### 9.2.2.6 FLEXCANx Clock

The FLEXCAN uses the DIVH as the input source of its functional clock. Also, the FLEXCAN implements an extra clock source which selects the clock from the bus clock.

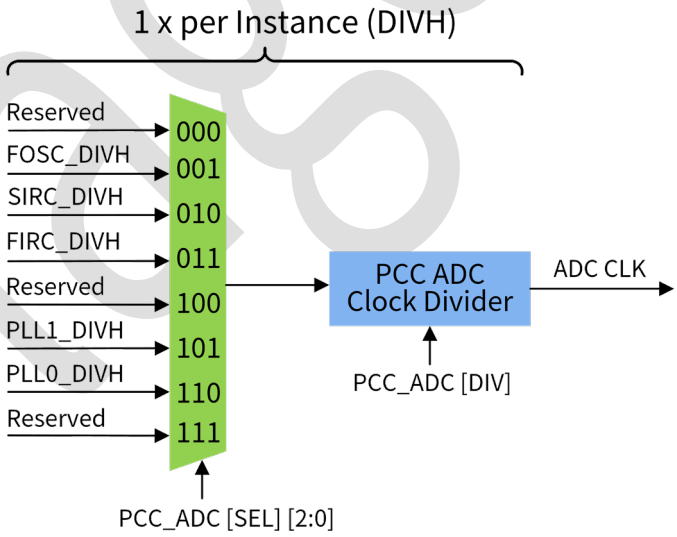
**Figure 16. FLEXCAN PE clock**




#### 9.2.2.7 ADCx Functional Clock

The ADC uses the DIVH as the input source of its functional clock.

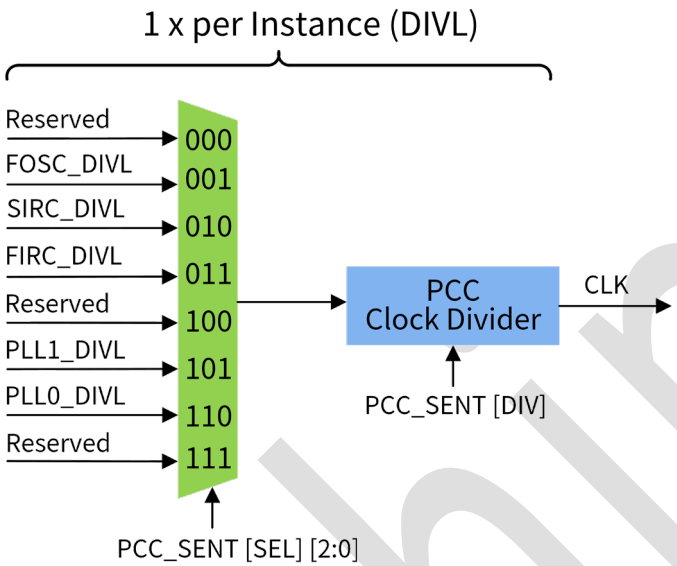
**Figure 17. ADC functional clock**



#### 9.2.2.8 SENT Clock

The SENT clock uses DIVL as the input source of its functional clock.

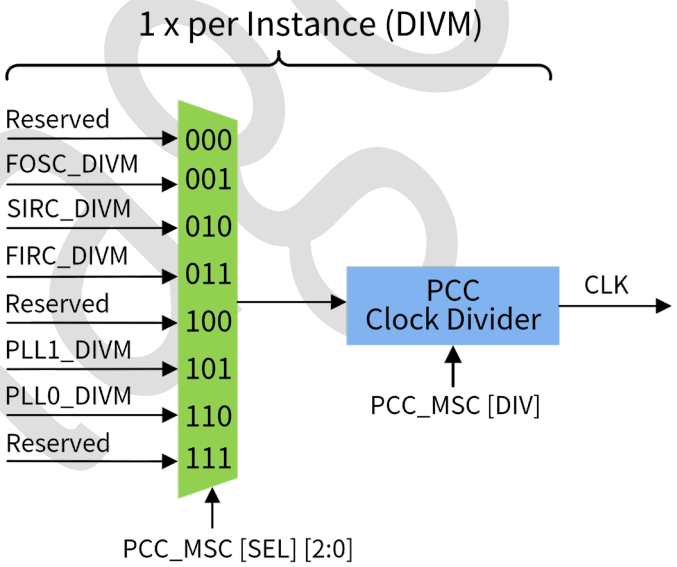
**Figure 18. SENT functional clock**




#### 9.2.2.9 MSC Clock

The MSC clock uses DIVM as the input source of its functional clock.

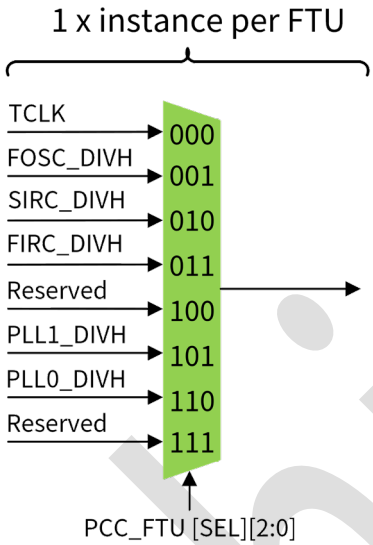
**Figure 19. MSC functional clock**



#### 9.2.2.10 FTUx TCLK Input

The FTUx TCLK input comes from the pin or PCC clock mux. See the below diagram.

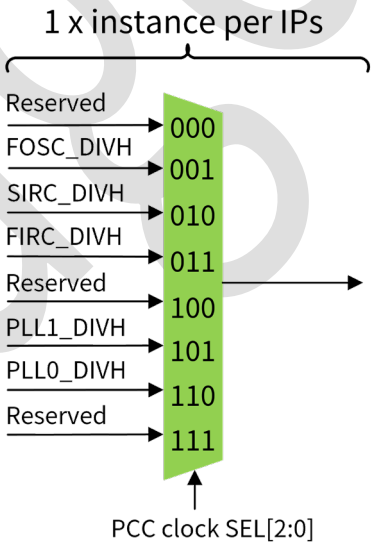
**Figure 20. FTUx functional clock input**




#### 9.2.2.11 FCPITx Functional Clock

The functional clock of FCPITx uses the DIVH as the input clock source.

**Figure 21. FCPITx functional clock**

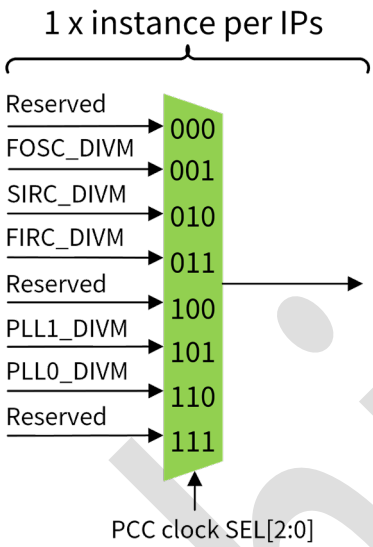



Flagchip

#### 9.2.2.12 FCSPIx/FCUARTx/FCIICx etc. Functional Clock

The functional clocks of FCSPIx/FCUARTx/FCIICx etc. use the DIVM as the input clock source.

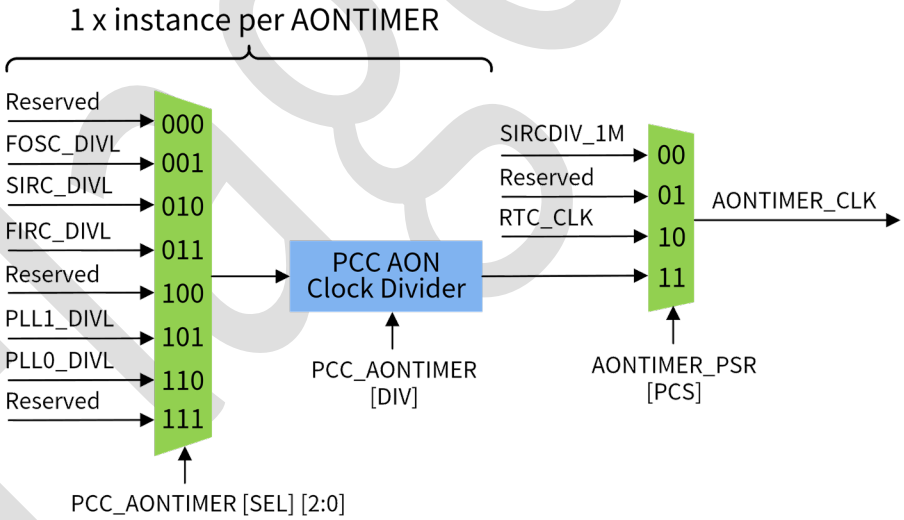
**Figure 22. IP functional clock**




#### 9.2.2.13 AONTIMER Clock

The AONTIMER contains a PCC divider, and it uses the DIVL as the root clock source.

**Figure 23. AONTIMER0 clock**



# Chapter 10 Reset and Boot Overview

This chapter summarizes the FC7240F2M reset and boot architecture.

## 10.1 Reset/Reset Generation Module (RGM)

The RGM contains the reset pin filter control, record of different reset sources' status, reset interrupt generation etc.

The FC7240F2M has the following reset sources:

**Table 31. Reset sources**

| Reset Type | Reset Sources |
|---|---|
| Power-on Reset (POR) | POR |
| System Reset | External pin reset |
| | JTAG reset |
| | Low-Voltage Reset (LVR) |
| | Clock Error 0 reset |
| | Clock Error 1 reset |
| | STOP entry timeout reset |
| | FCSMU error reset |
| | SYS-AP system reset |
| | HSM error reset |
| | CMU3 and CMU4 monitor failure reset |
| | Logic Built-In Self-Test (LBIST) request |
| | Reset monitor error |
| | PIN reset timeout |
| | System reset timeout |

Moreover, each core has its own reset sources:

- Software reset
- Lockup reset
- INTMx timeout reset
- WDOGx reset

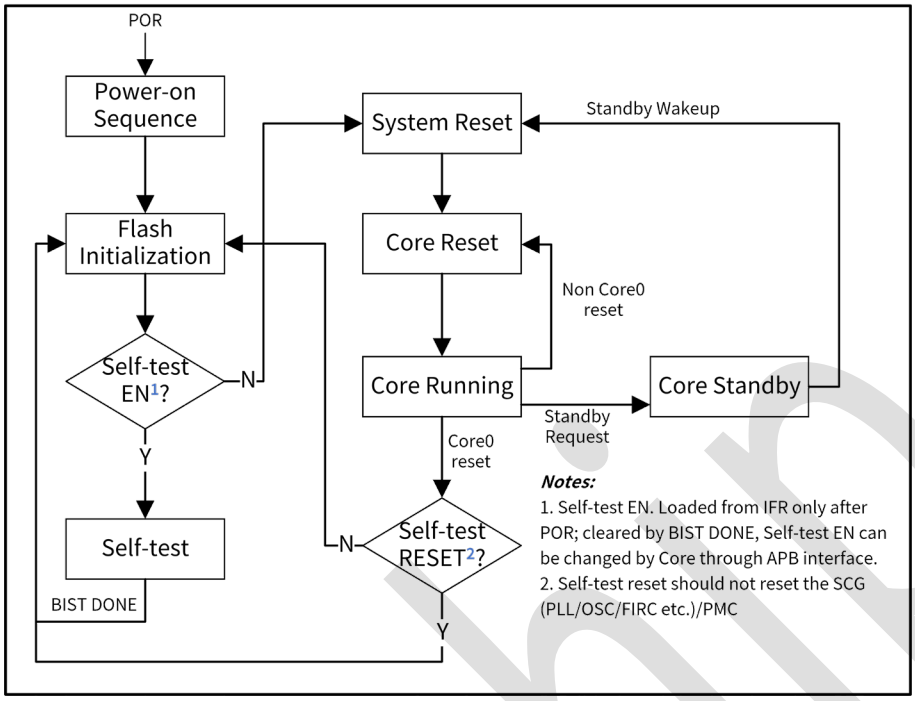
A CPU0 reset always also triggers a system reset.

For the PD1 domain, there is also a low power wakeup reset, which will reset the PD1 domain logic when wakeup from Standby mode.

***Note:*** The LBIST operation can only be triggered after a POR or an LBIST request triggered by software, depending on configuration. For a pin reset, the user can optionally enable the LBIST.

Flagchip

**Figure 24. Reset sequence**




The RGM can optionally generate a reset interrupt, so CPU can enter the pre-determined state.

## 10.2 Boot

The FC7240F2M supports secure boot and non-secure boot part. The NVR contains the secure boot/ In-System Program (ISP) control.

WDOG0 is enabled by default. The HOST ROM supports the debug challenge/response through SYS_AP. When waking up from standby/non-POR reset etc., the secure boot signature check can be optionally bypassed (depending on configuration) to achieve fast startup.

**Figure 25. Boot flow**




To support the ISP function, the ISP pin is added. To avoid entering the ISP unintentionally, the ISP pin is only sampled when RESET_b pin is asserted for enough time. The ISP pin status is stored in the SCM. The boot ROM can read the SCM to decide whether to enter ISP mode during boot.

Besides, the user can use the debug mailbox to send the ISP command to let the chip enter the ISP mode.

The FC7240F2M uses FCUART1/FCUART5/FLEXCAN1/FLEXCAN3 for ISP communication. The default pins of these instances are as follows.

**Table 32. ISP communication instances and pins**

| Instance | TX Pin | RX Pin |
|---|---|---|
| FCUART1 | PTC7 | PTC6 |
| FCUART5 | PTC15 | PTC16 |
| FLEXCAN1 | PTA13 | PTA12 |
| FLEXCAN3 | PTD7 | PTD6 |

Located on the pad PTD2, the ISP pin ISP_EN_B is disabled by default, and it can be enabled in the NVR.

Flagchip

# Chapter 11 Power Architecture

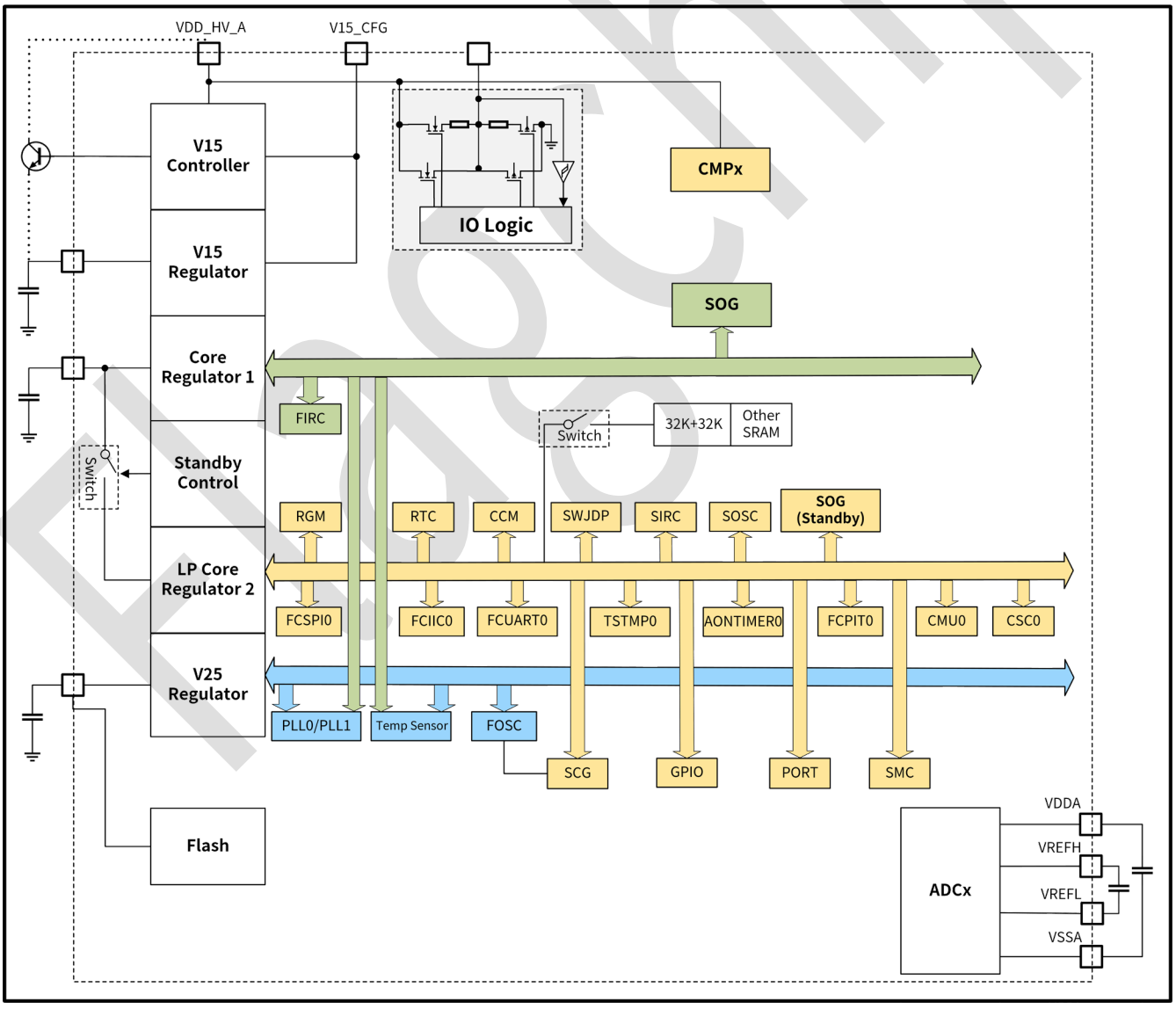
This chapter summarizes the FC7240F2M power architecture.

## 11.1 Power Connection Diagram

The below figure is the FC7240F2M power domain partition and power connection overview. Users can select to use an external Negative-Positive-Negative (NPN) transistor to drive the V15, or an external Power Management Integrated Circuit (PMIC) to drive the V15 directly (V15_CFG can be tied to VDD_HV_A or GND through a 1K resistor, and if V15_CFG is tied to VDD_HV_A, it means using the external PMIC to drive V15 directly). If using the external PMIC to drive V15, the V15 should be powered up after the VDD_HV_A supply reaches 2.3 V.

***Note:*** V15 Regulator is selected by default after power-up if V15_CFG is tied to ground. Software can switch V15 Regulator to V15 Controller (using external BJT) after power-up. Users can configure the pin V15_CFG to VDD, so that the external PMIC can be used to supply V15 directly. In this case, the V15_BASE_DRIVER can be used as GPIO also.

**Figure 26. FC7240F2M power diagram**



## 11.2 Power Mode

- RUN
- WAIT
- STOP
- STANDBY

## 11.3 Power Domain

The FC7240F2M supports 2 power domains (PD), including RUN domain (PD1), and AON or Standby domain (PD0).

Only pins located in the PD0 segment can keep working during standby mode. The PD1 pin segment will isolate to previous state. After reset, the user needs reconfigure the pin related function.

The FC7240F2M only supports 1 HV segment, and both VDD_HV_A and VDD_HV_B should be of the same supply.

The standby domain contains the below functions.

- All GPIOs, part of IOMUX
- Clock sources such as SIRC12M, SIRC32K, and SOSC
- CMP0-1
- CMU0-3, PCC0 (standby IPs) and CSC0-2
- FCUART0/FCSPI0/FCIIC0
- AONTIMER0/RTC/FCPIT0/TSTMP0
- Debug/SWJDP/JTAG etc.
- PMC/RGM/SMC/WKU
- STCU

The V25 supply is used by Flash, PLL0, PLL1, SOSC, Temperature sensor, FIRC etc. The V25 supply can be optionally on.

The V15 input is used as the internal V11 FPM LDO supply.

V11 is used for the RUN domain.

V11 standby is used for the Standby power domain.

The V15 is shut off during Standby mode. After POR/reset, the V15 will select the internal V15 LDO by default (if V15_CFG is tied to ground). Users can switch to the external BJT later via software.

## 11.4 Power Management Controller (PMC)/System Mode Controller (SMC)

- Support required supply generation such as V15, V11, and V25
- Support the power supply range monitor on VDD_HV_A, VDD_HV_B, V15, V11, V25 etc., and able to generate resets such as LVR, LVD, and HVD
- Support the POR generation
- Support the bandgap (BG) and low-power BG generation
- Support the low power mode switch control

### 11.4.1 Exteranl SBC/PMIC Interface

To support using the MCU signal output to wake up external SBC/PMIC, the system provides the LP_WAKEUP (refer to pin out) output. The LP_WAKEUP source can come from the RTC alarm output, AONTIMER time out, CMP output and SMC LP state control output. The LP_WAKUP should be able to output an active high/low (polarity) during Standby mode.

LP_WAKEUP also supports PWM outputs, uses FCPIT0 CH0 trigger as a PWM set signal, and CH1 trigger as a PWM clear signal (before polarity).

Flagchip

# Chapter 12 Security

## 12.1 NVR Control (Debug etc.)

The FC7240F2M supports secure boot. The debug security control relies on the Non-volatile Region (NVR) setting in the Flash configuration table. The default register setting allows operations including mass erase, debug unlock, and test mode entry. Refer to *Chapter SEC* for a detailed register introduction.

**Table 33. FC7240F2M NVR configuration assignments**

| NVR Configuration Name | Auto Load | Start Address (hex) | End Address (hex) | MSB | Description | Comments |
|---|---|---|---|---|---|---|
| NVRVLD | | 000 | 007 | | | 0x0000_0000_FC35_03DF |
| | | 008 | 00F | 63 | | |
| System security Control0_0 (FSC0) | | 010 | 017 | 127 | Non-Secure, FSC0[15:0] = 0xC35A & FSC1[15:0] = 0xFFFF<br>Secure, Other values | Only available in non-secure boot mode. It has no effect in secure boot mode. |
| System security Control0_1 (FSC0) | | 018 | 01F | 191 | | |
| System security Control1_0 (FSC1) | | 020 | 027 | 255 | | |
| System security Control1_1 (FSC1) | | 028 | 02F | 319 | | |
| Debug Control0 (DC0) | | 030 | 037 | 383 | • Bit 3:0 – NEP (NVR Erase Protection, 0x5/0xF no protection)<br>• Bit 7:4 – DBA (Debug Backdoor Allow, 0x5 allow backdoor function)<br>• Bit 11:8 – RWL (Register Write Lock, 0x5/0xF allow register write)<br>• Bit 15:12 – TME (Test Mode Enable, 0x5/0xF enable)<br>• Bit 19:16 – TMBA (Test Mode Backdoor Allow, 0x5/0xF enable)<br>• Bit 23:20 – MED (Mass Erase Disable, 0x5/0xF enable)<br>• Bit 27:24 – NKRP (NVR key read protection, 0x5/0xF no protection)<br>Bit 31:28 – NWP (NVR program protection, 0x5/0xF no protection) | |

Flagchip

**Table 33. FC7240F2M NVR configuration assignments (continued)**

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | • Bit 63:32 Reserved | |
| Debug Control1 (DC1) | | 038 | 03F | 447 | Bit 3:0 – UKAE (User key access enable, 0x5/0xF enable) | SEC registers (RO) |
| | | 040 | 047 | 511 | | |
| | | 048 | 04F | 575 | | |
| BOOT Configuration 0 (BC0) | | 050 | 057 | 639 | • Bit 0 – Fast boot select<br>0 – Select PLL0 as Core clock source, core clock is 240MHz<br>1 – Select FIRC as Core clock source, core clock is 96MHz<br>• Bit 1 – NMI function enable/disable<br>0 – NMI pin function is enabled, the associated pin default to internal pull up enabled.<br>1 – NMI pin function is disabled after reset<br>• Bit 2 – Reserved<br>• Bit 3 – BOOTROM<br>0 – Boot from non-ROM<br>1 – Boot from ROM<br>• Bit 63:60 – INSTANCEID[3:0] | |
| BOOT Configuration 1 (BC1) | | 058 | 05F | 703 | BOOT Source | |
| BOOT Configuration 2 (BC2) | | 060 | 067 | 767 | • Bit 12:0 – Reserved<br>• Bit 31:13 – CPU0 BootAddr<br>• Bit 32 – Reserved<br>• Bit 33 – Reserved<br>• Bit 39:34 – Reserved<br>• Bit 41:40 – AXBS ECC ENABLE<br>• Bit 43:42 – CPU0 ECC ENABLE<br>• Bit 45:44 – Reserved<br>• Bit 47:46 – Reserved<br>• Bit 63:48 – Reserved | |
| BOOT Configuration 3 (BC3) | | 068 | 06F | 831 | • Bit 12:0 – Reserved<br>• Bit 31:13 – Reserved<br>• Bit 44:32 – Reserved<br>• Bit 63:45 – Reserved | |
| BootROM Configuration 0 (BRC0) | | 070 | 077 | 895 | • Bit 3:0 – Reserved<br>• Bit 7:4 – host_debug_auth_en (0xF or 0x5 means enable, other values mean disable) | SEC registers (RO) |

**Table 33. FC7240F2M NVR configuration assignments (continued)**

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | • Bit 11:8 – ISP_auth_en (0xF or 0x5 means enable, other values mean disable) | |
| Reserved | | 078 | 07F | 959 | | |
| BootROM Configuration 1 (BRC1) | | 080 | 087 | 1023 | • Bit 3:0 – Reserved<br>• Bit 5:4 – FCUART baud rate for ISP<br>2'b00 – 2Mbps<br>2'b01 – 1Mbps<br>2'b10 – 512Kbps<br>2'b11 – 115200bps<br>• Bit 7:6 – FLEXCAN baud rate for ISP<br>2'b00 – CANFD 4Mbps/CAN 1Mbps if CANFD is not valid.<br>2'b01 – CANFD 2Mbps/CAN 1Mbps if CANFD is not valid.<br>2'b10 – CAN 1Mbps<br>2'b11 – CAN 500Kbps<br>• Bit 10:8 – OSC frequency<br>3'b000 – 8MHz<br>3'b001 – 12MHz<br>3'b010 – 16MHz<br>3'b011 – 24MHz<br>3'b100 – 25MHz<br>3'b101 – 32MHz<br>3'b110 – 48MHz<br>3'b111 – 40MHz<br>• Bit 11 – OSC available<br>1'b0 – OSC is available<br>1'b1 – OSC is not available<br>• Bit 15:12 – DBK input enable<br>4'b0000 – user cannot input DBK through both Debug Mailbox and ISP;<br>4'b0011 – user can input DBK through Debug Mailbox, ISP is not valid;<br>4'b1100 – user can input DBK through ISP, Debug mailbox is not valid;<br>4'b1111 – user can input DBK through both Debug Mailbox and ISP;<br>Other values – user cannot input DBK through both Debug Mailbox and ISP.<br>• Bit 16 – User code verification enable in non-secure boot mode | SEC registers (RO) |

Flagchip

**Table 33. FC7240F2M NVR configuration assignments (continued)**

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | 1'b0 – User code verification enable in non-secure boot mode<br>1'b1 – User code verification disable in non-secure boot mode<br>• Bit 19:17 – Debug backdoor key encryption algorithm:<br>3'b000 – AES256-CBC<br>3'b111 – SM4-CBC<br>• Bit 20 – ISP Pin disable/enable (Only valid in non-secure mode)<br>1'b0 – ISP pin is enabled during pin reset<br>1'b1 – ISP pin is disabled<br>• Bit 21, ISP mode allow in recovery mode<br>1'b0 – ISP mode is not allowed in recovery mode<br>1'b1 – ISP mode is allowed in recovery mode<br>• Bit 22 – Enable the output of plaintext decrypted ROOT key<br>1'b0 – The output of plaintext decrypted ROOT key is enabled<br>1'b1 – The output of plaintext decrypted ROOT key is disabled<br>• Bit 23, ROM loop control flag<br>1'b0 – ROM executes normal run in VIRGIN and FT state.<br>1'b1 – ROM enters debug loop in VIRGIN and FT state.<br>• Bit 24 – Enable software configuration to enter ISP<br>1'b0 – Software configuration to enter ISP is enabled.<br>1'b1 – Software configuration to enter ISP is disabled.<br>• Bit 63:32, Reserved | |
| Reserved | | 088 | 08F | 1087 | | |

Flagchip

**Table 33. FC7240F2M NVR configuration assignments (continued)**

| | | | | | | |
|---|---|---|---|---|---|---|
| BootROM Configuration 2 (BRC2) | | 090 | 097 | 1151 | • Bit 24:0 – usercode_verify_mask, {C1_C2_reset, RGM.SRS[31:29], RGM.C0_SRS[20:0]}<br>• Bit 27:25 – ISP instance select<br>3'b000 – FCUART1<br>3'b001 – FCUART5<br>3'b010 – FLEXCAN1<br>3'b011 – FLEXCAN3<br>3'b111 – No selection, should poll all the instances<br>• Bit 30:28 – usercode_verification_alg<br>3'b000 – AES-CMAC (only available when FlexCore enable)<br>3'b001 – ECC256 + SHA256<br>3'b010 – SM4-CMAC (only available when FlexCore enable)<br>3'b011 – RSA4096 (only available when FlexCore enable)<br>3'b111 – SM2 + SM3<br>• Bit 31, Debug/ISP/PRE_FA Authentication and USRK decryption algorithm<br>1'b0 – ECC256<br>1'b1 – SM2<br>• Bit 63:34 – image_address (always 32-bit aligned) | SEC registers (RO) |
| NVR Version | | 098 | 09F | 1215 | • Bit 23:0 – NVR_VER<br>• Bit 63:32 – Reserved | SEC registers (RO) |
| Reserved | | 0A0 | 0A7 | 1279 | | |
| Reserved | | 0A8 | 0AF | 1343 | | |
| LBIST Test Ctrl | | 0B0 | 0B7 | 1407 | • Bit 31:0 – self_test_ctrl0<br>• Bit 63:32 – self_test_ctrl1 | |
| MBIST Test Ctrl | | 0B8 | 0BF | 1471 | • Bit 31:0 – mbist_sel<br>• Bit 63:32 – mbist_alg | |
| LBIST Ctrl | | 0C0 | 0C7 | 1535 | • Bit 31:0 – lbist_ctrl0<br>• Bit 63:32 – lbist_ctrl1 | |
| LBIST MISR | | 0C8 | 0CF | 1599 | • Bit 31:0 – lbist_misr0<br>• Bit 63:32 – lbist_misr1 | |
| | | 0D0 | 0D7 | 1663 | | |
| | | 0D8 | 0DF | 1727 | | |

**Table 33. FC7240F2M NVR configuration assignments (continued)**

| | | | | | | |
|---|---|---|---|---|---|---|
| FlexCore Enable | | 0E0 | 0E7 | 1791 | • Bit 3:0 – FlexCore_En (1111 and 0101 mean disable, others mean enable)<br>• Bit 31:4 – Reserved<br>• Bit 63:34 – Code address for the FlexCore (always 4-byte aligned) | |
| PFLASH parallel Enable | | 0E8 | 0EF | 1855 | • Bit 3:0 – PFLASH_parallel_En (1111 means enable, 0101 means disable, others mean enable)<br>• Bit 63:4 – Reserved | |
| | | 0F0 | 0F7 | 1919 | | |
| | | 0F8 | 0FF | 1983 | | |
| OTA Configure 0 (OTAC0) | | 100 | 107 | 2047 | • Bit 4:0 – OTA enable control, 01010 means OTA enabled, others means OTA disable<br>• Bit 53:32 – OTA version location in PFLASH (always 128-bit aligned, 35:32 bits values will be ignored by hardware) | |
| Reserved | | 108 | 10F | 2111 | | |
| Reserved | | 110 | 117 | 2175 | | |
| Reserved | | 118 | 11F | 2239 | | |
| OTA Configure 2 (OTAC2) | | 120 | 127 | 2303 | • Bit 4:0 – OTA enable control, 01010 means OTA enabled, others mean OTA disable<br>• Bit 53:32 – OTA version location in PFLASH (always 128-bit aligned, 35:32 bits values will be ignored by hardware) | |
| Reserved | | 128 | 12F | 2367 | | |
| Reserved | | 130 | 137 | 2431 | | |
| Reserved | | 138 | 13F | 2495 | | |
| Secure Boot Enable | | 140 | 147 | 2559 | Bit 3:0 – secure_boot_en | |
| Reserved | | 148 | 14F | 2623 | | |
| Secure Boot Enable Backup | | 150 | 157 | 2687 | Bit 3:0 – secure_boot_en_backup | |
| Reserved | | 158 | 15F | 2751 | | |
| Lifecycle 1_0 | | 160 | 167 | 2815 | • Bit 31:0 – OEM_DEV mark value 0<br>• Bit 63:32 – OEM_DEV mark value 1 | Lifecycle1: OEM_DEV |
| Lifecycle 1_1 | | 168 | 16F | 2879 | | Lifecycle1: OEM_DEV |
| Lifecycle 1_2 | | 170 | 177 | 2943 | | Lifecycle1: |

**Table 33. FC7240F2M NVR configuration assignments (continued)**

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | OEM_DEV |
| Lifecycle 1_3 | | 178 | 17F | 3007 | • Bit 31:0 – OEM_DEV mark value 2<br>• Bit 63:32 – OEM_DEV mark value 3 | Lifecycle1: OEM_DEV |
| Lifecycle 2_0 | | 180 | 187 | 3071 | • Bit 31:0 – OEM_PDT mark value 0<br>• Bit 63:32 – OEM_PDT mark value 1 | Lifecycle2: OEM_PDT |
| Lifecycle 2_1 | | 188 | 18F | 3135 | | Lifecycle2: OEM_PDT |
| Lifecycle 2_2 | | 190 | 197 | 3199 | | Lifecycle2: OEM_PDT |
| Lifecycle 2_3 | | 198 | 19F | 3263 | • Bit 31:0 – OEM_PDT mark value 2<br>• Bit 63:32 – OEM_PDT mark value 3 | Lifecycle2: OEM_PDT |
| Lifecycle 3_0 | | 1A0 | 1A7 | 3327 | • Bit 31:0 – IN_FIELD mark value 0<br>• Bit 63:32 – IN_FIELD mark value 1 | Lifecycle3: IN_FIELD |
| Lifecycle 3_1 | | 1A8 | 1AF | 3391 | | Lifecycle3: IN_FIELD |
| Lifecycle 3_2 | | 1B0 | 1B7 | 3455 | | Lifecycle3: IN_FIELD |
| Lifecycle 3_3 | | 1B8 | 1BF | 3519 | • Bit 31:0 – IN_FIELD mark value 2<br>• Bit 63:32 – IN_FIELD mark value 3 | Lifecycle3: IN_FIELD |
| Lifecycle 4_0 | | 1C0 | 1C7 | 3583 | • Bit 31:0 – SWFA mark value 0<br>• Bit 63:32 – SWFA mark value 1 | Lifecycle4: SWFA |
| Lifecycle 4_1 | | 1C8 | 1CF | 3647 | | Lifecycle4: SWFA |
| Lifecycle 4_2 | | 1D0 | 1D7 | 3711 | | Lifecycle4: SWFA |
| Lifecycle 4_3 | | 1D8 | 1DF | 3775 | • Bit 31:0 – SWFA mark value 2<br>• Bit 63:32 – SWFA mark value 3 | Lifecycle4: SWFA |
| | | 1E0 | 1E7 | 3839 | | |
| | | 1E8 | 1EF | 3903 | | |
| | | 1F0 | 1F7 | 3967 | | |
| | | 1F8 | 1FF | 4031 | | |
| MRK0 Revoke | | 200 | 207 | 4095 | Bit 31:0 – mrk_revoke0 | |
| Reserved | | 208 | 20F | 4159 | | |
| MRK1 Revoke | | 210 | 217 | 4223 | Bit 31:0 – mrk_revoke1 | |
| Reserved | | 218 | 21F | 4287 | | |
| MRK2 Revoke | | 220 | 227 | 4351 | Bit 31:0 – mrk_revoke2 | |
| Reserved | | 228 | 22F | 4415 | | |
| MRK3 Revoke | | 230 | 237 | 4479 | Bit 31:0 – mrk_revoke3 | |
| Reserved | | 238 | 23F | 4543 | | |
| UMRK0 Revoke | | 240 | 247 | 4607 | Bit 31:0 – umrk_revoke0 | |
| Reserved | | 248 | 24F | 4671 | | |

**Table 33. FC7240F2M NVR configuration assignments (continued)**

| | | | | | | |
|---|---|---|---|---|---|---|
| UMRK1 Revoke | | 250 | 257 | 4735 | Bit 31:0 – umrk_revoke1 | |
| Reserved | | 258 | 25F | 4799 | | |
| UMRK2 Revoke | | 260 | 267 | 4863 | Bit 31:0 – umrk_revoke2 | |
| Reserved | | 268 | 26F | 4927 | | |
| UMRK3 Revoke | | 270 | 277 | 4991 | Bit 31:0 – umrk_revoke3 | |
| Reserved | | 278 | 27F | 5055 | | |
| Lifecycle 5_0 | | 280 | 287 | 5119 | • Bit 31:0 – HWFA mark value 0<br>• Bit 63:32 – HWFA mark value 1 | Lifecycle5: HWFA |
| Lifecycle 5_1 | | 288 | 28F | 5183 | | Lifecycle5: HWFA |
| Lifecycle 5_2 | | 290 | 297 | 5247 | | Lifecycle5: HWFA |
| Lifecycle 5_3 | | 298 | 29F | 5311 | • Bit 31:0 – HWFA mark value 2<br>• Bit 63:32 – HWFA mark value 3 | Lifecycle5: HWFA |
| | | 2A0 | 2A7 | 5375 | | |
| | | 2A8 | 2AF | 5439 | | |
| | | 2B0 | 2B7 | 5503 | | |
| | | 2B8 | 2BF | 5567 | | |
| | | 2C0 | 2C7 | 5631 | | |
| | | 2C8 | 2CF | 5695 | | |
| | | 2D0 | 2D7 | 5759 | | |
| | | 2D8 | 2DF | 5823 | | |
| | | 2E0 | 2E7 | 5887 | | |
| | | 2E8 | 2EF | 5951 | | |
| | | 2F0 | 2F7 | 6015 | | |
| | | 2F8 | 2FF | 6079 | | |
| Clock Reconfig Flag | | 300 | 307 | 6143 | Clock config valid flag | |
| Clock Reconfig Flag Mask | | 308 | 30F | 6207 | if value & mask == 0xC001BEEF_FC2011FE, SCG/PCC/CAN/UART config is valid | |
| SCG CCR REG | | 310 | 317 | 6271 | Bit 31:0 – SCG_CCR | |
| SCG FOSC0 | | 318 | 31F | 6335 | • Bit 31:0 – SCG_FOSC_CSR,<br>• Bit 63:32 – SCG_FOSC_DIV | |
| SCG FOSC1 | | 320 | 327 | 6399 | Bit 31:0 – SCG_FOSC_CFG | |
| SCG FIRC 0 | | 328 | 32F | 6463 | • Bit 31:0 – SCG_FIRC_CSR,<br>• Bit 63:32 – SCG_FIRC_DIV | |
| SCG FIRC 1 | | 330 | 337 | 6527 | • Bit 31:0 – SCG_FIRC_CFG<br>• Bit 63:32 – SCG_FIRC_TCFG | |
| SCG SPLL0_0 | | 338 | 33F | 6591 | • Bit 31:0 – SCG_SPLL0_CSR,<br>• Bit 63:32 – SCG_SPLL0_DIV | |
| SCG SPLL0_1 | | 340 | 347 | 6655 | Bit 31:0 – SCG_SPLL0_CFG | |

**Table 33. FC7240F2M NVR configuration assignments (continued)**

| | | | | | | |
|---|---|---|---|---|---|---|
| SCG SPLL1_0 | | 348 | 34F | 6719 | • Bit 31:0 – SCG_SPLL1_CSR,<br>• Bit 63:32 – SCG_SPLL1_DIV | |
| SCG SPLL1_1 | | 350 | 357 | 6783 | Bit 31:0 – SCG_SPLL1_CFG | |
| PCC | | 358 | 35F | 6847 | • Bit 31:0 – PCC_CAN,<br>• Bit 63:32 – PCC_UART | |
| FLEXCAN Config 0 | | 360 | 367 | 6911 | • Bit 31:0, CAN Ctrl1,<br>• Bit 63:32, FDCTRL | |
| FLEXCAN Config 1 | | 368 | 36F | 6975 | • Bit 31:0, CBT,<br>• Bit 63:32, FDCBT | |
| UART Config | | 370 | 377 | 7039 | Bit 31:0 – UART BAUD | |
| System clk freq | | 378 | 37F | 7103 | • Bit 31:0 – Core_clk_freq,<br>• Bit 63:32 – Bus_clk_freq | |
| Reserved | | 380 | 77F | 15295 | Reserved | Reserved |
| Asym_Public_Key_HASH 0_1 | | 780 | 787 | 15359 | • Bit 31:0 – APKH0<br>• Bit 63:32 – APKH1 | |
| Asym_Public_Key_HASH 2_3 | | 788 | 78F | 15423 | • Bit 31:0 – APKH2<br>• Bit 63:32 – APKH3 | |
| Asym_Public_Key_HASH 4_5 | | 790 | 797 | 15487 | • Bit 31:0 – APKH4<br>• Bit 63:32 – APKH5 | |
| Asym_Public_Key_HASH 6_7 | | 798 | 79F | 15551 | • Bit 31:0 – APKH6<br>• Bit 63:32 – APKH7 | |
| CMAC_Key 0_1 | | 7A0 | 7A7 | 15615 | • Bit 31:0 – CMK0<br>• Bit 63:32 – CMK1 | |
| CMAC_Key 2_3 | | 7A8 | 7AF | 15679 | • Bit 31:0 – CMK2<br>• Bit 63:32 – CMK3 | |
| CMAC_Key 4_5 | | 7B0 | 7B7 | 15743 | • Bit 31:0 – CMK4<br>• Bit 63:32 – CMK5 | |
| CMAC_Key 6_7 | | 7B8 | 7BF | 15807 | • Bit 31:0 – CMK6<br>• Bit 63:32 – CMK7 | |
| Block Cipher Key 0/1 | | 7C0 | 7C7 | 15871 | • Bit 31:0 – BCK0<br>• Bit 63:32 – BCK1 | |
| Block Cipher Key 2/3 | | 7C8 | 7CF | 15935 | • Bit 31:0 – BCK2<br>• Bit 63:32 – BCK3 | |
| Block Cipher Key 4/5 | | 7D0 | 7D7 | 15999 | • Bit 31:0 – BCK4<br>• Bit 63:32 – BCK5 | |
| Block Cipher Key 6/7 | | 7D8 | 7DF | 16063 | • Bit 31:0 – BCK6<br>• Bit 63:32 – BCK7 | |
| TestMode Overwrite Key | | 7E0 | 7E7 | 16127 | Bit 31:0 – TMBK (TestMode Overwrite Key) | |
| | | 7E8 | 7EF | 16191 | | |
| Debug | | 7F0 | 7F7 | 16255 | • Bit 31:0 – DBK0 | In non-secure |

**Table 33. FC7240F2M NVR configuration assignments (continued)**

| | | | | | | |
|---|---|---|---|---|---|---|
| Backdoor key0/1 | | | | | • Bit 63:32 – DBK1 | boot mode, BCK can be used to unlock debug and ISP commands; in secure boot mode, BCK cannot used to unlock debug but can be used to unlock ISP commands. |
| Debug Backdoor key2/3 | | 7F8 | 7FF | 16319 | • Bit 31:0 – DBK2<br>• Bit 63:32 – DBK3 | |

The following table shows the OTA content, users need to follow the requirement in order to activate the version.

**Table 34. OTA content**

| | | | | | | |
|---|---|---|---|---|---|---|
| OTA indicator in PFLASH0 (128:64) | | | | | only valid if equal to 64'hFC20_FACE_8888_6666 | Valid code |
| OTA indicator in PFLASH0 (63:0) | | | | | if all FF, invalid OTA indicator<br>only if 63:32 = ~31:0 valid OTA indicator<br>15:0 represent the software version, hardware/software should select the valid PFLASH0 or PFLASH1 with higher version as active source | |

For the Lifecycle, the following table shows the valid code.

**Table 35. Valid code for the lifecycle**

| **Lifecycle[255:192]** | **Lifecycle[63:0]** | **Status** |
|---|---|---|
| Erased | Erased | Erased |
| Erased | Marked | Active |
| Marked | Marked | Inactive |
| Other | Other | Illegal |

FA cycle is only allowed to be entered after passing the Flagchip authentication flow. Refer to Chapter *SEC* for detailed secure/debug related settings.

## 12.2 HSM Subsystem Hardware

The HSM on the FC7240F2M supports functions including AES/SHA/ECC/RSA and Random Number and is able to support algorithm acceleration such as SM2/SM3/SM4/SM9. Besides, the HSM supports the key storage management, including key import and export. The host can trigger the HSM service through the Mailbox interrupt channel 1.

For the HSM firmware, which can be installed to NVR0 and NVR1 (16KB for each; 2KB for each sector. The first 2KB of NVR0 is for the User NVR configuration). For the HSM key, which can be installed to NVR2 (16KB in total; 8 sectors).

Flagchip

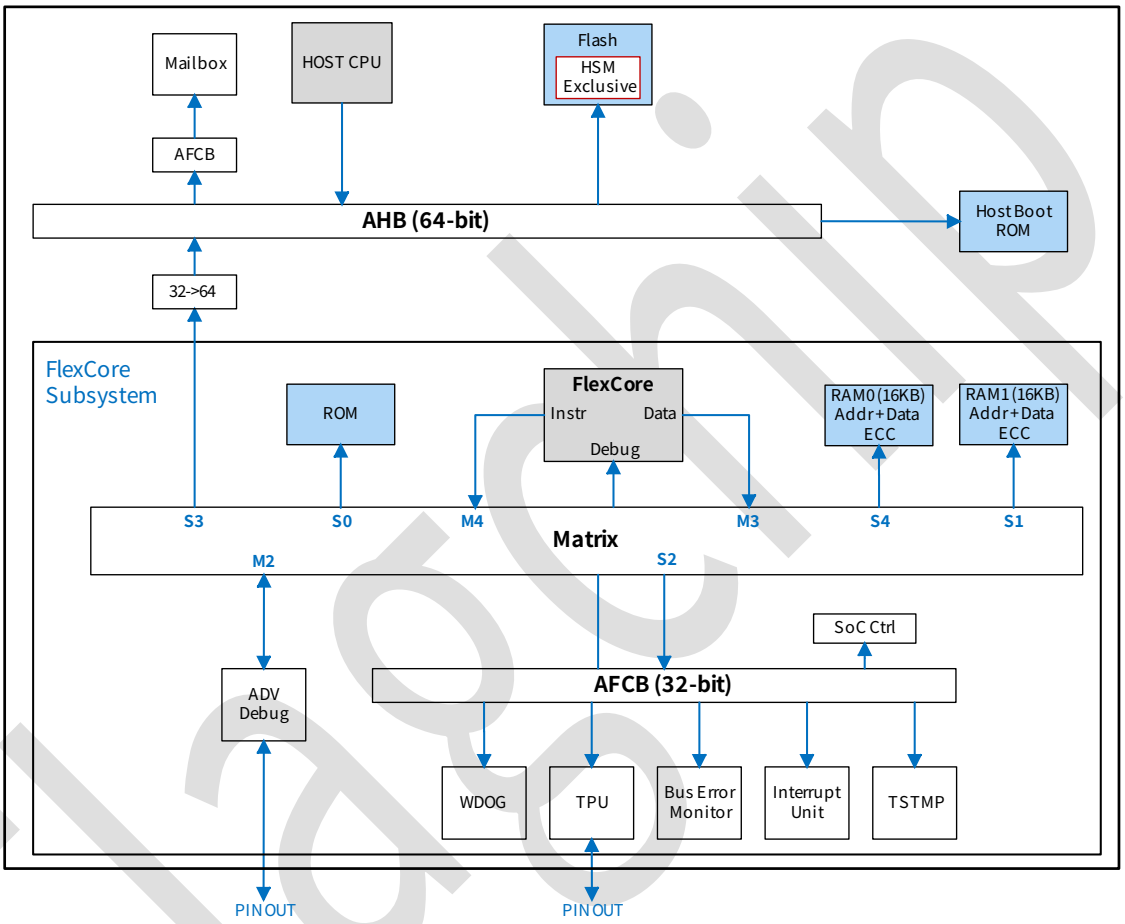
# Chapter 13 FlexCore Mode

This chapter introduces the FlexCore mode-related information in FC7240F2M.

***Note:*** In FlexCore mode, the subsystem refers to the FlexCore; in non-FlexCore mode, the subsystem refers to the HSM.

The following figure illustrates the overall block diagram of the FlexCore subsystem.

**Figure 27. Overall FlexCore subsystem block diagram**




For details of the FlexCore subsystem, refer to *FC7240 FlexCore Subsystem Reference Manual.*

## 13.1 Instruction Sets Supported by FlexCore

- Full support for RV32I Base Integer Instruction Set
- Full support for RV32C Standard Extension for Compressed Instructions
- Full support for RV32M Integer Multiplication and Division Instruction Set Extension

## 13.2 FlexCore Memory

The following table shows the FlexCore memory map.

**Table 36. FlexCore memory map**

| Start Address (hex) | End Address (hex) | Approx. Size | Modules | Comments |
|---|---|---|---|---|
| 0000_0000 | 0001_7FFF | 96 KB | FlexCore BOOTROM | Including the FlexCore self-test code |
| 0001_8000 | 0001_FFFF | 32 KB | Reserved | |

**Table 36. FlexCore memory map (continued)**

| Start Address (hex) | End Address (hex) | Approx. Size | Modules | Comments |
|---|---|---|---|---|
| 0002_0000 | 0002_3FFF | 16 KB | FlexCore instruction RAM | 16 KB for FlexCore subsystem instruction RAM |
| 0002_4000 | 0002_7FFF | 16 KB | FlexCore data RAM | 16 KB for FlexCore subsystem data RAM |
| 0002_8000 | 00FF_FFFF | | Reserved | |
| 0100_0000 | 011F_FFFF | 2048 KB | PFlash for Access PORT0 | 2048KB for each PORT |
| 0120_0000 | 017F_FFFF | 6144 KB | Reserved | |
| 0180_0000 | 027F_FFFF | 4*4096 KB | Reserved | Reserved Program Flash for PORTx |
| 0400_0000 | 0401_FFFF | 128 KB | DFlash | |
| 0402_0000 | 043F_FFFF | | Reserved | |
| 0440_0000 | 0440_07FF | 2 KB | NVR | |
| 0440_0800 | 0440_3FFF | 14 KB | Reserved | |
| 0440_8000 | 0440_BFFF | 16 KB | Reserved | |
| 0441_0000 | 0441_1FFF | 8 KB | Reserved | |
| 0442_0000 | 0442_07FF | 2 KB | Reserved | |
| 0480_0000 | 0481_FFFF | 128 KB | Host ROM | |
| 0800_0000 | 080F_FFFF | 1024 KB | Reserved | |
| 1000_0000 | 1FFF_FFFF | 256 MB | Reserved for external memory | |
| 2000_0000 | 2001_FFFF | 128 KB | DTCM | |
| 2100_0000 | 2100_FFFF | 64 KB | SRAM_0 | |
| 2101_0000 | 2101_7FFF | 32 KB | SRAM_1 | |
| 2104_0000 | 2107_FFFF | 256 KB | Reserved | |
| 2200_0000 | 2201_FFFF | 128 KB | Reserved | |
| 4000_0000 | 401F_FFFF | 2048 KB | AFCB0 | |
| 4040_0000 | 405F_FFFF | 2048 KB | AFCB1 | |
| 4100_0000 | 41FF_FFFF | 16 MB | AHB ports/RGPIO | 64 KB/per AHB port |
| 4200_0000 | 423F_FFFF | 4096 KB | FlexCore AFCB | FlexCore subsystem AFCB peripherals |
| 4240_0000 | 5FFF_FFFF | | Reserved | |
| 6700_0000 | 67FF_FFFF | 128 MB | Reserved | |
| 6800_0000 | 6FFF_FFFF | 128 MB | Reserved | |
| E000_0000 | E00F_FFFF | 1024 KB | Reserved | |

## 13.3 FlexCore AFCB

The following table shows the FlexCore AFCB slot assignments.

**Table 37. FlexCore AFCB**

| Peripheral Description | Peripheral Instance | SEG0 Slot | Size (KB) | Start Address (hex) | End Address (hex) |
|---|---|---|---|---|---|
| Reserved | Reserved | 0~31 | 128 | 4200_0000 | 4201_FFFF |
| Time Processing Unit | TPU | 32 | 4 | 4202_0000 | 4202_0FFF |

**Table 37. FlexCore AFCB (continued)**

| Peripheral Description | Peripheral Instance | SEG0 Slot | Size (KB) | Start Address (hex) | End Address (hex) |
|---|---|---|---|---|---|
| Engine | | | | | |
| Watchdog | WDOG | 33 | 4 | 4202_1000 | 4202_1FFF |
| Interrupt Unit | INTU | 34 | 4 | 4202_2000 | 4202_2FFF |
| Subsystem Control | SS_CTRL | 35 | 4 | 4202_3000 | 4202_3FFF |
| Timer Stamp | TSTMP | 36 | 4 | 4202_4000 | 4202_4FFF |
| Reserved | Reserved | 37 | 4 | 4202_5000 | 4202_5FFF |
| Simple Direct Memory Access Controller | SDMAC | 38 | 4 | 4202_6000 | 4202_6FFF |
| Bus Error Monitor | BEM | 39 | 4 | 4202_7000 | 4202_7FFF |
| Reserved | Reserved | 40 | 4 | 4202_8000 | 4202_8FFF |
| | | 41 | 4 | 4202_9000 | 4202_9FFF |
| | | 42 | 4 | 4202_A000 | 4202_AFFF |
| | | 43 | 4 | 4202_B000 | 4202_BFFF |
| Instruction Memory Protection Unit | IMPU | 44 | 4 | 4202_C000 | 4202_CFFF |
| Data Memory Protection Unit | DMPU | 45 | 4 | 4202_D000 | 4202_DFFF |
| Reserved | Reserved | 46~47 | 16 | 4202_E000 | 4202_FFFF |
| Debug Port | DBG | 48~55 | 32 | 4203_0000 | 4203_7FFF |
| Reserved | Reserved | 56~63 | 24 | 4203_8000 | 4203_FFFF |

## 13.4 FlexCore Interrupt

The following table shows FlexCore interrupt connection.

**Table 38. FlexCore interrupt assignments**

| IRQ Index | Interrupt Source | INT Vector Address |
|---|---|---|
| 0 | RGM_RESET_IRQ | 0x0000_0000 |
| 1 | BUS_ERR_IRQ | 0x0000_0004 |
| 2 | IRAM_ECC_IRQ | 0x0000_0008 |
| 3 | DRAM_ECC_IRQ | 0x0000_000C |
| 4 | TPU_IRQ | 0x0000_0010 |
| 5 | TPU_HSR_IRQ | 0x0000_0014 |
| 6 | Reserved | 0x0000_0018 |
| 7 | Mailbox_IRQ | 0x0000_001C |
| 8 | TSTMP_IRQ | 0x0000_0020 |
| 9 | DMA_IRQ | 0x0000_0024 |
| 10 | WDOG_IRQ | 0x0000_0028 |
| 11 | STOP_REQ_IRQ | 0x0000_002C |
| 12 | FLASH_ALT_IRQ | 0x0000_0030 |

**Table 38. FlexCore interrupt assignments (continued)**

| IRQ Index | Interrupt Source | INT Vector Address |
|---|---|---|
| 13 | FLEXCAN3_IRQ | 0x0000_0034 |
| 14 | FLEXCAN2_IRQ | 0x0000_0038 |
| 15 | FTU0_IRQ | 0x0000_003C |
| 16 | FTU1_IRQ | 0x0000_0040 |
| 17 | ADC0_IRQ | 0x0000_0044 |
| 18 | ADC1_IRQ | 0x0000_0048 |
| 19 | GPIOA_IRQ | 0x0000_004C |
| 20 | GPIOB_IRQ | 0x0000_0050 |
| 21 | GPIOC_IRQ | 0x0000_0054 |
| 22 | GPIOD_IRQ | 0x0000_0058 |
| 23 | GPIOE_IRQ | 0x0000_005C |
| 24 | CMP0_IRQ | 0x0000_0060 |
| 25 | CMP1_IRQ | 0x0000_0064 |
| 26 | IMPU_IRQ | 0x0000_0068 |
| 27 | DMPU_IRQ | 0x0000_006C |
| 28 | | 0x0000_0070 |
| 29 | | 0x0000_0074 |
| 30 | | 0x0000_0078 |
| 31 | | 0x0000_007C |

## 13.5 FlexCore SRAM

The FlexCore IRAM and DRAM support the 32 + 8 ECC structure, which is able to detect single-bit address ECC errors and double-bit data ECC errors, and able to perform single-bit data ECC correction. The EIM provides the IRAM/DRAM ECC function injection test capability.

## 13.6 FlexCore Control

For details on control of the FlexCore mode, see descriptions of SEC_FLEXCORE_EN in *Chapter SEC.*

## 13.7 TPU-related Control

The TPU-related control information varies depending on if in the FlexCore mode.

**Table 39. TPU-related control**

| TPU-related Control | FlexCore Mode | Non-FlexCore Mode |
|---|---|---|
| **Clock** | Enabled by Host CPU (CM7) & Engine CPU (FlexCore) | Enabled by Host/Engine CPU (CM7) |
| **Reset** | Controlled by Host CPU (CM7) or Engine CPU (FlexCore) | Controlled by Host/Engine CPU (CM7) |
| **Debug** | Depending on if Engine CPU is in debug | Depending on if Host/Engine CPU (CM7) is in debug |

# Chapter 14 Function Safety Overview

## 14.1 Introduction

The FC7240F2M family is developed following the ISO 26262 standards, with derivatives targeting the chips that are operable in a system that fulfills the requirements of the ASIL D or ASIL B safety integrity levels.

The ASIL level targets the safety processing, which means the software functions are executed as expected:

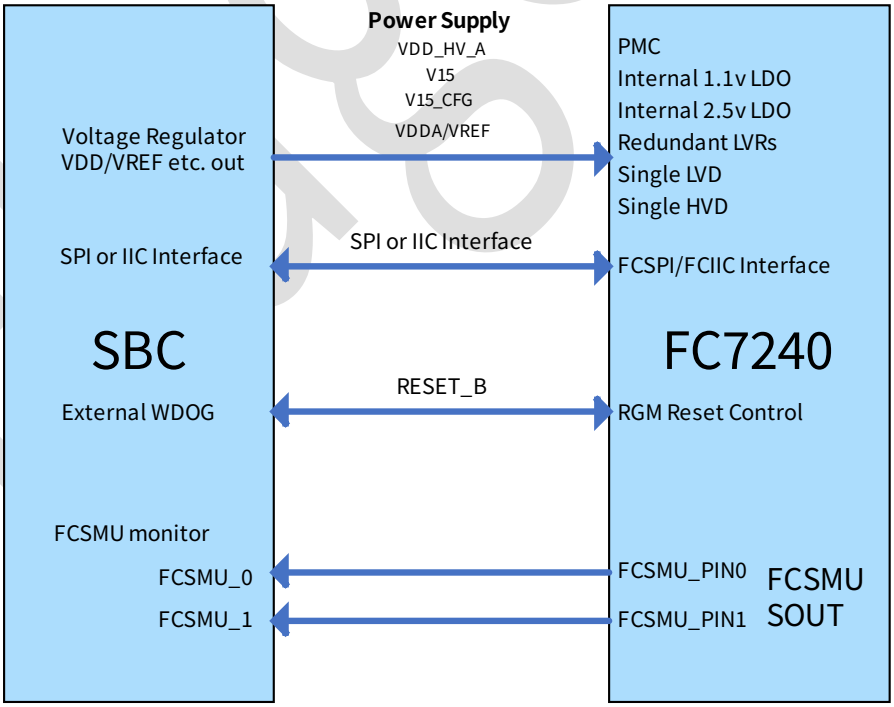
1) Read instructions from memories such as Flash/TCM/ROM.
2) Execute instructions by Cortex_M7.
3) Read data from memories such as TCM or System RAM.
4) Process data by Cortex_M7.
5) Write back the data into memories such as TCM or System RAM.

The CM7_0 achieves ASIL D compliance. For peripherals, their ASIL levels depend on the level of the system or application they are going to be integrated with. Most peripherals are multi-instanced and with separate pins/AFCB slots.

It assumes that the chip is connected to an external System Basis Chip (SBC) or Power Management Integrated Circuit (PMIC) to achieve ASIL D compliance on the power supply. The SBC or PMIC performs observations and control functionalities that are essential to fulfill some related functional safety requirements. If no SBC or PMIC is used, the user must ensure that the system provides an equivalent functionality that properly manages the corresponding interface(s) and function(s).

The following figure illustrates the safety interface between the chip and the SBC or PMIC.

**Figure 28. Safety interface between the FC7240 and SBC/PMIC**



The FC7240F2M supports a function safety level of ASIL B and is compliant with AEC-Q100/Grade 1. It has the following hardware safety-related features.

- A system reset brings all pads to the high-z state (MCU safe state). Only JTAG_TCK/TMS/TDI share pads and RESET/NMI share pads are pull enabled. If ISP_EN_b function is enabled, it will enable the internal pullup function when the reset pin is held low.
- Reset/NMI pins support the analog glitch filter function, avoiding generating a false reset or NMI event. Reset pins also support the digital glitch filter function.
- Pin input with the digital glitch filter function, avoiding generating a false pin interrupt event.
- Reset FSM monitor, which monitors the RGM reset activity. Unintended FSM switching will trigger a system reset.
- Timeout monitor, which monitors the system standby enter/exit flow.
- Pin and system RESET monitor. If a pin request or system reset request asserts above the specific time, it will force to generate a system reset.
- Two WDOGs for CM7 core, with independent clock sources and the window refresh capability. Certain WDOG registers support a lock option to avoid the settings being changed by the software unintentionally. The WDOG can be used to monitor the software execution flow.
- The FCSMU collects the on-chip faults and can be configured to generate an interrupt or a reset. Besides, the FCSMU output pins can be used to work with an external SBC.
- The FCSMU has an internal CRC logic to monitor the FCSMU configuration register.
- The STCU contains the LBIST control and result capture, as well as the MBIST for SRAM and ROM BIST tests.
- CPU0 supports the lockstep function with lockstep monitors, covering CM7, ITCM ECC, DTCM0&1 ECC, AXI2AHB gasket, and other functions.
- Support End to End ECC check for all masters, such as checks for address/data, and transfer attributes. Support the 64 to 32 downsize monitor. Support the S2F gasket monitor.
- Clock monitors. CMU0 - 3 monitors the clock root such as FIRC, SIRC, SOSC, and FOSC. CMU4 is used to monitor the slow clock frequency. Any clock output beyond the preconfigured range can trigger an interrupt.
- Flash controller with ECC protection. An ECC error will trigger a bus fault to the system. The flash controller ECC supports both address and data error detection.
- System SRAM controller with ECC protection; able to support both address and data ECC errors, as well as support single-bit correction and double-bit detection. An ECC error will trigger an interrupt to the system. Also, SRAM data read (pass to MAM) is re-encoded and compared with previous ECC check bits to make sure that SRAM ECC logic working as expected, and any mismatch will trigger a fault to the FCSMU.
- ITCM RAM with ECC protection; able to support both address and data ECC errors, as well as support single-bit correction and double-bit detection. An ECC error will trigger a bus fault to the system.
- DTCM0 and DTCM1 RAM with ECC protection; able to support both address and data ECC error, as well as support single-bit correction and double-bit detection. An ECC error will trigger a bus fault to the system.
- AFCB0/1 with monitor. All AFCB0/1 accesses are monitored.
- FLEXCAN mailbox RAM with ECC protection; able to support single-bit correction and double-bit detection. An ECC error will trigger a bus fault to the system.
- DMACFG RAM with ECC protection; able to support single-bit correction and double-bit detection, as well as support both address and data ECC. An ECC error will trigger an ERM interrupt to the system.
- The DMA0 implements the lockstep function. Any error will trigger an error to the FCSMU.
- All CPUs' cache with ECC protection on both ICACHE and DCACHE.
- ROM with ECC protection. With address/data ECC error protection.

- EIM/ERM. ECC error injection modules for the on-chip ECC decoding logics. For the ECC error information related to ECC error report modules, refer to EIM/ERM for detailed channel mapping.
- The INTM can trigger a reset or an interrupt (depending on settings) if the preselected interrupt is not serviced in the preconfigured period.
- The FTU with fault inputs, which can put an FTU output into a safe state.
- HSM IRAM and DRAM support both address and data ECC, which is able to support single-bit correction and double-bit detection. Moreover, HSM IRAM/DRAM data read (pass to FlexCore) is re-encoded and compared with previous ECC check bits to make sure that SRAM ECC logic working as expected, and any mismatch will trigger a fault to Host.
- The PMC implements supply monitors, including POR on VDD_HV_A and LVR/LVD/HVD on supplies.
- Multiple VDD_HV/VSS pins to avoid package failure of a single pin on the pinout.
- Signals, including V15, V11, V25, VDDA, and VBG, are connected to an ADC channel, so the system can double-check the power supply via the ADC.
- The CM7 supports the 16-region MPU function.
- The MAM0 provides the on-chip resource access check/protection. It also monitors the matrix access with watchdog function.
- Flash Memory Controller supports the flash program/erase protection function.
- Hardware CRC, which can be used by software for CRC calculation for modules such as memory and communication.
- The TMU monitors the chip junction temperature.
- CMP DAC output to the ADC for the ADC self-test check.
- ROM API for the CM7 Core self-test check which can be used for ASIL B core.
- The NVR loading implements an NVR monitor to monitor the NVR loading procedure.
- Flash Controller supports hardware program/erase flow monitor, which avoids unintentionally programming/erasing the flash content. In addition, there is a program/erase timeout monitor to make sure the program and erase HV operations do not exceed the limit.
- The ISM monitors the period, duty, and delay of critical input and output signals. It supports up to 16 input monitors.
- Triple registers are implemented in IPs' critical flip flops.

The FC7240F2M also implements two on-chip peripheral matrixes, which allow using spare FCSPI/FCUART to monitor other FCSPI/FCUART's transmit status.

- The SCM block includes an internal matrix that allows using any FCSPIx to monitor other FCSPIx's transmit interface. FCSPIx here indicates FCSPI0 – 5.
- The SCM block includes an internal matrix that allows using any FCUARTx to monitor other FCUARTx's transmit interface. FCUARTx here indicates FCUART0 - 7.

For the LBIST/MBIST, it supports two test ways from the application level.

1) Power up test (offline test), which tests as much as possible within the required startup time. The LBIST test vector number, MBIST test algorithm, and other parameters can be configured and loaded from the NVR.
2) Power down test or diagnostic test (online test), with the maximum test coverage and no test time limitation. The LBIST test and MBIST test configuration can be set by software. After programming the STCU, software can trigger a software reset to perform an STCU test.

Except for the non-LBIST modules such as SCG, RGM, PMC, and STCU, the rest blocks are all under LBIST.

The PTD20, PTA2 and PTA3 can be used to indicate the LBIST internal status when entering the LBIST state. Refer to *Chapter STCU* for detailed settings.

## 14.2 Safety IP Overview

### 14.2.1 FCSMU Channel Assignments

The FCSMU collects different system-level faults and decides the fault reaction. The below table lists the FCSMU channel assignments. It can be configured to generate an interrupt or a reset.

**Table 40. FCSMU channel assignments**

| FCSMU Channels | | |
|---|---|---|
| **Channel Number** | **Failure** | **Failure Description** |
| 0 | TEMP_ERROR | Event from temperature sensor |
| 1 | PMC_ERROR | Voltage out of range indication from PMC |
| 2 | MISC_ERROR0 | NVR load error/System abnormal alarm signal, including MBIST active and Test Mode active |
| 3 | STCU_BIST_FAIL | STCU MBIST or LBIST fail |
| 4 | STCU LS0 ERR | Lockstep compare fault (including CPU, ITCM lockstep error, DTCM lockstep error, and CPU overlay error) |
| 5 | Reserved | Reserved |
| 6 | SYSRAM_CPU0_ERR | System RAM CPU0 access error |
| 7 | Reserved | Reserved |
| 8 | Reserved | Reserved |
| 9 | SYSRAM_NON_CPU_ERR | System RAM None CPU access error |
| 10 | SCM_CPU0_ERR | Matrix Access Monitor ECC check CPU0 error (including CPU AHBP monitor error) |
| 11 | Reserved | Reserved |
| 12 | Reserved | Reserved |
| 13 | SCM_NON_CPU_ERR | Matrix Access Monitor ECC check non CPU error (including AHBS S2F gasket monitor error and downsize gasket monitor error) |
| 14 | CPU0_ECC_ERR | Including ITCM, DTCM, ICACHE, and DCACHE |
| 15 | Reserved | Reserved |
| 16 | Reserved | Reserved |
| 17 | Reserved | Reserved |
| 18 | Reserved | Reserved |
| 19 | CMU4 failure interrupt | CMU4 failure interrupt |
| 20 | CMU failure interrupt | CMU1/2 failure interrupt |
| 21 | FLASH ECC Error | Flash ECC error |
| 22 | SCG_PLL_FOSC | PLL/FOSC loss of clock |
| 23 | DMA0_ERR | Lockstep Error/CFG RAM Error |
| 24 | Reserved | Reserved |
| 25 | INTM0 | Interrupt Monitor error |
| 26 | Reserved | Reserved |
| 27 | Reserved | Reserved |
| 28 | MISC_ERROR1 | FMC_ECC_ERROR/AFCB0&1 monitor error/SRAMx EDC error/HSM error (including SRAM ECC and WDOG) |

**Table 40. FCSMU table assignment (continued)**

| FCSMU Channels | | |
|---|---|---|
| **Channel Number** | **Failure** | **Failure Description** |
| 29 | MISC_ERROR2 | SCG and SCM etc. CRC check error |
| 30 | Reserved | Reserved |
| 31 | MAM WDOG TIME OUT | MAM master access time out error |

### 14.2.2 STCU

- Support the LBIST and MBIST control. Support NVR load and CPU access for control registers.
- Support lockstep functions such as FSM/Count.
- Support WDOG during LBIST/MBIST. If a WDOG timeout is triggered, it will abort the LBIST/MBIST.
- Support initializing the ITCM/DTCM/SRAM by STCU.

### 14.2.3 EIM/ERM

#### *14.2.3.1 EIM Channel Assignments*

The EIM provides the error injection capability related to all on-chip ECC decoding logics. The below table lists the EIM channel assignments.

**Table 41. EIM channel assignments**

| EIM Bank Channels | |
|---|---|
| **Channel Number** | **Source Description** |
| 0 | SCM Matrix Access Monitor Error Injection0 (MAM0_S0) |
| 1 | SCM Matrix Access Monitor Error Injection1 (MAM0_S1) |
| 2 | SCM Matrix Access Monitor Error Injection2(MAM0_S2) |
| 3 | Reserved |
| 4 | SCM Matrix Access Monitor Error Injection3 (MAM0_S3) |
| 5 | SCM Matrix Access Monitor Error Injection4 (MAM0_S4) |
| 6 | Reserved |
| 7 | SCM Matrix Access Monitor Error Injection5 (MAM0_S5) |
| 8 | SCM Matrix Access Monitor Error Injection6 (MAM0_S6) |
| 9 | SCM Matrix Access Monitor Error Injection7 (MAM0_S7) |
| 10 | SCM Matrix Access Monitor Error Injection8 (MAM0_S8) |
| 11 | SCM Matrix Access Monitor Error Injection9 (CPU0_AHBM) |
| 12 | SCM Matrix Access Monitor Error Injection10 (CPU0_AHBP) |
| 13-17 | Reserved |
| 18 | SCM Matrix Access Monitor Error Injection12 (HSM) |
| 19 | SCM Matrix Access Monitor Error Injection13 (DMA0) |
| 20 | Reserved |
| 21 | SCM Matrix Access Monitor Error Injection11 (CPU0_AHBS) |
| 22 | Reserved |
| 23 | Reserved |
| 24 | CPU0_ITCM (including address) |
| 25 | CPU0_DTCM0 (including address) |

**Table 41. EIM channel assignment (continued)**

| **EIM Bank Channels** | |
|---|---|
| **Channel Number** | **Source Description** |
| 26 | CPU0_DTCM1 (including address) |
| 27-32 | Reserved |
| 33 | SRAM0_INJECTION (including address) |
| 34 | SRAM1_INJECTION (including address) |
| 35 | Reserved |
| 36 | CPU0_DCACHE_DATA0_01 Injection |
| 37 | CPU0_DCACHE_DATA0_23 Injection |
| 38 | CPU0_DCACHE_DATA1_01 Injection |
| 39 | CPU0_DCACHE_DATA1_23 Injection |
| 40 | CPU0_DCACHE_TAG_01 Injection |
| 41 | CPU0_DCACHE_TAG_23 Injection |
| 42 | CPU0_ICACHE_DATA0 Injection |
| 43 | CPU0_ICACHE_DATA1 Injection |
| 44 | CPU0_ICACHE_TAG Injection |
| 45-62 | Reserved |
| 63 | SCM S5 64:32 DownSize Monitor Error Injection |
| 64 | Reserved |
| 65 | DMA0_CFG_ECC Injection (including address) |
| 66 | Reserved |
| 67 | ROM_ECC Injection (including address) |
| 68 | SCM S5 DownSize ECC Check with AHBS Error Injection |
| 69 | Reserved |
| 70 | SCM CPU0 AHBP and AHBS gasket Monitor Error Injection |
| 71-74 | Reserved |
| 75 | SCM S8 64:32 DownSize Monitor Error Injection |
| 76 | Reserved |
| 77 | RAM Decoder Monitor Error Injection |
| 78 | AFCB0 Monitor Error Injection |
| 79 | AFCB1 Monitor Error Injection |
| 80 | CPU0 Overlay Monitor |
| 81 | Reserved |
| 82 | HSM DRAM ECC Error Injection |
| 83 | HSM IRAM ECC Error Injection |
| **EIM Lockstep Channels** | |
| 0 | CPU0_LOCKSTEP Injection (including ITCM/DTCM/AXI2AHB) |
| 1 | Reserved |
| 2 | DMA0_LCOKSTEP injection |

Flagchip

#### 14.2.3.2 ERM Channel Assignments

The below table lists the ERM channel assignments.

**Table 42. ERM channel assignments**

| ERM channels | |
|---|---|
| **Channel Number** | **Source Description** |
| 0 | Reserved |
| 1 | PFlash0 ECC Error |
| 2 | PFlash1 ECC Error |
| 3 | DFlash ECC error |
| 4 | DMACFG0 ECC Error |
| 5 | Reserved |
| 6 | ROM ECC Error |
| 7 | Reserved |
| 8 | SysRAM0 ECC ERROR |
| 9 | SysRAM1 ECC ERROR |
| 10 | Reserved |
| 11 | CPU0 ITCM ECC Error |
| 12 | CPU0 DTCM0 ECC Error |
| 13 | CPU0 DTCM1 ECC Error |
| 14 | CPU0 ICACHE ECC Error |
| 15 | CPU0 DCACHE ECC Error |
| 16-25 | Reserved |
| 26 | HSM DRAM ECC Error |
| 27 | HSM IRAM ECC Error |

#### 14.2.3.3 EIM Channel Error Injection and Check

The table lists the corresponding channel configuration and error check for different EIM channels.

**Table 43. EIM matrix error check (bank channels)**

| **EIM Bank Channels** | | **EIM BUS Register Field Configuration** | | | | **Error Check** | | |
|---|---|---|---|---|---|---|---|---|
| **CH. No.** | **Source Description** | **EIM_BUS_REG0[DATA0] Field Value** | **EIM_BUS_REG0[DATA1] Field Value** | **EIM_BUS_REG0[ADDR] Field Value** | **EIM_BUS_REG0[ATTR] Field Value** | **SCM Matrix Status Register** | **FCSMU CH.** | **ERM CH.** |
| 0 | SCM Matrix Access Monitor Error Injection0 (MAM0_S0) | 0~71 | 0~71 | N/A | N/A | **Slave End Error:** N/A<br>**CPU0 Master End Error:**<br>STATUS2[C0_AHBM_D]<br>STATUS2[C0_AHBM_M]<br>**DMA0 Master End Error:**<br>STATUS1[DMA0_D]<br>STATUS1[DMA0_M]<br>**HSM Master End Error:** | CPU0: CH_10<br>NONCPU: CH_13 | N/A |

**Table 43. EIM matrix error check (bank channels) (continued)**

| EIM Bank Channels | | EIM BUS Register Field Configuration | | | | Error Check | | |
|---|---|---|---|---|---|---|---|---|
| CH. No. | Source Description | EIM_BUS_REG0[DATA0] Field Value | EIM_BUS_REG0[DATA1] Field Value | EIM_BUS_REG0[ADDR] Field Value | EIM_BUS_REG0[ATTR] Field Value | SCM Matrix Status Register | FCSMU CH. | ERM CH. |
| | | | | | | STATUS1[HSM_D]<br>STATUS1[HSM_M] | | |
| 1 | SCM Matrix Access Monitor Error Injection1 (MAM0_S1) | 0~71 | 0~71 | 3~26 | 0~23 | **Slave End Error:**<br>STATUS0[MAM0_S1_A]<br>STATUS0[MAM0_S1_D]<br>STATUS0[MAM0_S1_AT]<br>STATUS0[MAM0_S1_M]<br>**CPU0 Master End Error:**<br>STATUS2[C0_AHBM_A]<br>STATUS2[C0_AHBM_D]<br>STATUS2[C0_AHBM_M]<br>**DMA0 Master End Error:**<br>STATUS1[DMA0_A]<br>STATUS1[DMA0_D]<br>STATUS1[DMA0_M]<br>**HSM Master End Error:**<br>STATUS1[HSM_A]<br>STATUS1[HSM_D]<br>STATUS1[HSM_M] | CPU0: CH_10<br>NONCPU: CH_13 | N/A |
| 2 | SCM Matrix Access Monitor Error Injection2 (MAM0_S2) | 0~71 | 0~71 | 3~26 | 0~23 | **Slave End Error:**<br>STATUS0[MAM0_S2_A]<br>STATUS0[MAM0_S2_D]<br>STATUS0[MAM0_S2_AT]<br>STATUS0[MAM0_S2_M]<br>**CPU0 Master End Error:**<br>STATUS2[C0_AHBM_A]<br>STATUS2[C0_AHBM_D]<br>STATUS2[C0_AHBM_M]<br>**DMA0 Master End Error:**<br>STATUS1[DMA0_A]<br>STATUS1[DMA0_D]<br>STATUS1[DMA0_M]<br>**HSM Master End Error:**<br>STATUS1[HSM_A] | CPU0: CH_10<br>NONCPU: CH_13 | N/A |

Flagchip

**Table 43. EIM matrix error check (bank channels) (continued)**

| EIM Bank Channels | | EIM BUS Register Field Configuration | | | | Error Check | | |
|---|---|---|---|---|---|---|---|---|
| CH. No. | Source Description | EIM_BUS_REG0[DATA0] Field Value | EIM_BUS_REG0[DATA1] Field Value | EIM_BUS_REG0[ADDR] Field Value | EIM_BUS_REG0[ATTR] Field Value | SCM Matrix Status Register | FCSMU CH. | ERM CH. |
| | | | | | | STATUS1[HSM_D]<br>STATUS1[HSM_M] | | |
| 3 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 4 | SCM Matrix Access Monitor Error Injection4 (MAM0_S3) | 0~71 | 0~71 | 3~15 | 0~23 | **Slave End Error:**<br>STATUS0[MAM0_S3_A]<br>STATUS0[MAM0_S3_D]<br>STATUS0[MAM0_S3_AT]<br>STATUS0[MAM0_S3_M]<br>**CPU0 Master End Error:**<br>STATUS2[C0_AHBM_A]<br>STATUS2[C0_AHBM_D]<br>STATUS2[C0_AHBM_M]<br>**DMA0 Master End Error:**<br>STATUS1[DMA0_A]<br>STATUS1[DMA0_D]<br>STATUS1[DMA0_M]<br>**HSM Master End Error:**<br>STATUS1[HSM_A]<br>STATUS1[HSM_D]<br>STATUS1[HSM_M] | CPU0: CH_10<br>NONCPU: CH_13 | N/A |

Flagchip

**Table 43. EIM matrix error check (bank channels) (continued)**

| EIM Bank Channels | | EIM BUS Register Field Configuration | | | | Error Check | | |
|---|---|---|---|---|---|---|---|---|
| CH. No. | Source Description | EIM_BUS_REG0[DATA0] Field Value | EIM_BUS_REG0[DATA1] Field Value | EIM_BUS_REG0[ADDR] Field Value | EIM_BUS_REG0[ATTR] Field Value | SCM Matrix Status Register | FCSMU CH. | ERM CH. |
| 5 | SCM Matrix Access Monitor Error Injection5 (MAM0_S4) | 0~71 | 0~71 | 3~15 | 0~23 | **Slave End Error:**<br>STATUS0[MAM0_S4_A]<br>STATUS0[MAM0_S4_D]<br>STATUS0[MAM0_S4_AT]<br>STATUS0[MAM0_S4_M]<br>**CPU0 Master End Error:**<br>STATUS2[C0_AHBM_A]<br>STATUS2[C0_AHBM_D]<br>STATUS2[C0_AHBM_M]<br>**DMA0 Master End Error:**<br>STATUS1[DMA0_A]<br>STATUS1[DMA0_D]<br>STATUS1[DMA0_M]<br>**HSM Master End Error:**<br>STATUS1[HSM_A]<br>STATUS1[HSM_D]<br>STATUS1[HSM_M] | CPU0: CH_10<br>NONCPU: CH_13 | N/A |
| 6 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 7 | SCM Matrix Access Monitor Error Injection7 (MAM0_S5) | 0~71 | 0~71 | 3~29 | 0~23 | **Slave End Error:**<br>STATUS0[MAM0_S5_A]<br>STATUS0[MAM0_S5_D]<br>STATUS0[MAM0_S5_AT]<br>STATUS0[MAM0_S5_M]<br>**CPU0 Master End Error:**<br>STATUS2[C0_AHBM_A]<br>STATUS2[C0_AHBM_D]<br>STATUS2[C0_AHBM_M]<br>**DMA0 Master End Error:**<br>STATUS1[DMA0_A]<br>STATUS1[DMA0_D]<br>STATUS1[DMA0_M]<br>**HSM Master End Error:**<br>STATUS1[HSM_A]<br>STATUS1[HSM_D]<br>STATUS1[HSM_M] | CPU0: CH_10<br>NONCPU: CH_13 | N/A |

Flagchip

**Table 43. EIM matrix error check (bank channels) (continued)**

| EIM Bank Channels | | EIM BUS Register Field Configuration | | | | Error Check | | |
|---|---|---|---|---|---|---|---|---|
| CH. No. | Source Description | EIM_BUS_REG0[DATA0] Field Value | EIM_BUS_REG0[DATA1] Field Value | EIM_BUS_REG0[ADDR] Field Value | EIM_BUS_REG0[ATTR] Field Value | SCM Matrix Status Register | FCSMU CH. | ERM CH. |
| 8 | SCM Matrix Access Monitor Error Injection8 (MAM0_S6) | 0~71 | 0~71 | 3~19 | 0~23 | **Slave End Error:**<br>STATUS1[MAM0_S6_A]<br>STATUS1[MAM0_S6_D]<br>STATUS1[MAM0_S6_AT]<br>STATUS1[MAM0_S6_M]<br>**CPU0 Master End Error:**<br>STATUS2[C0_AHBM_A]<br>STATUS2[C0_AHBM_D]<br>STATUS2[C0_AHBM_M]<br>**DMA0 Master End Error:**<br>STATUS1[DMA0_A]<br>STATUS1[DMA0_D]<br>STATUS1[DMA0_M]<br>**HSM Master End Error:**<br>STATUS1[HSM_A]<br>STATUS1[HSM_D]<br>STATUS1[HSM_M] | CPU0: CH_10<br>NONCPU: CH_13 | N/A |
| 9 | SCM Matrix Access Monitor Error Injection9 (MAM0_S7) | 0~71 | 0~71 | 3~19 | 0~23 | **Slave End Error:**<br>STATUS1[MAM0_S7_A]<br>STATUS1[MAM0_S7_D]<br>STATUS1[MAM0_S7_AT]<br>STATUS1[MAM0_S7_M]<br>**CPU0 Master End Error:**<br>STATUS2[C0_AHBM_A]<br>STATUS2[C0_AHBM_D]<br>STATUS2[C0_AHBM_M]<br>**DMA0 Master End Error:**<br>STATUS1[DMA0_A]<br>STATUS1[DMA0_D]<br>STATUS1[DMA0_M]<br>**HSM Master End Error:**<br>STATUS1[HSM_A]<br>STATUS1[HSM_D]<br>STATUS1[HSM_M] | CPU0: CH_10<br>NONCPU: CH_13 | N/A |

Flagchip

**Table 43. EIM matrix error check (bank channels) (continued)**

| EIM Bank Channels | | EIM BUS Register Field Configuration | | | | Error Check | | |
|---|---|---|---|---|---|---|---|---|
| CH. No. | Source Description | EIM_BUS_REG0[DATA0] Field Value | EIM_BUS_REG0[DATA1] Field Value | EIM_BUS_REG0[ADDR] Field Value | EIM_BUS_REG0[ATTR] Field Value | SCM Matrix Status Register | FCSMU CH. | ERM CH. |
| 10 | SCM Matrix Access Monitor Error Injection10(MAM0_S8) | 0~71 | 0~71 | 3~11 | 0~23 | **Slave End Error:**<br>STATUS1[MAM0_S8_A]<br>STATUS1[MAM0_S8_D]<br>STATUS1[MAM0_S8_AT]<br>STATUS1[MAM0_S8_M]<br>**CPU0 Master End Error:**<br>STATUS2[C0_AHBM_A]<br>STATUS2[C0_AHBM_D]<br>STATUS2[C0_AHBM_M]<br>**DMA0 Master End Error:**<br>STATUS1[DMA0_A]<br>STATUS1[DMA0_D]<br>STATUS1[DMA0_M]<br>**HSM Master End Error:**<br>STATUS1[HSM_A]<br>STATUS1[HSM_D]<br>STATUS1[HSM_M] | CPU0: CH_10<br>NONCPU: CH_13 | N/A |
| 11 | SCM Matrix Access Monitor Error Injection11 (CPU0_AHBM) | 0~71 | 0~71 | 3~30 | 0~23 | **Master End Error:**<br>STATUS2[C0_AHBM_A]<br>STATUS2[C0_AHBM_D]<br>STATUS2[C0_AHBM_M]<br>**MAM0_S3 Slave End Error:**<br>STATUS0[MAM0_S3_A]<br>STATUS0[MAM0_S3_D]<br>STATUS0[MAM0_S3_AT]<br>STATUS0[MAM0_S3_M] | CH_10 | N/A |
| 12 | SCM Matrix Access Monitor Error Injection12 (CPU0_AHBP) | 0~71 | 0~71 | 3~31 | 0~23 | **Master End Error:**<br>STATUS2[C0_AHBP_A]<br>STATUS2[C0_AHBP_D]<br>STATUS2[C0_AHBP_M]<br>**MAM0_S3 Slave End Error:**<br>STATUS0[MAM0_S3_A]<br>STATUS0[MAM0_S3_D]<br>STATUS0[MAM0_S3_AT]<br>STATUS0[MAM0_S3_M] | CH_10 | N/A |
| 13 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 14 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |

Flagchip

**Table 43. EIM matrix error check (bank channels) (continued)**

| EIM Bank Channels | | EIM BUS Register Field Configuration | | | | Error Check | | |
|---|---|---|---|---|---|---|---|---|
| CH. No. | Source Description | EIM_BUS_REG0[DATA0] Field Value | EIM_BUS_REG0[DATA1] Field Value | EIM_BUS_REG0[ADDR] Field Value | EIM_BUS_REG0[ATTR] Field Value | SCM Matrix Status Register | FCSMU CH. | ERM CH. |
| 15 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 16 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 17 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 18 | SCM Matrix Access Monitor Error Injection18 (HSM) | 0~71 | 0~71 | 3~31 | 0~23 | **Master End Error:**<br>STATUS1[HSM_A]<br>STATUS1[HSM_D]<br>STATUS1[HSM_M]<br>**HSM MAM0_S3 Slave End Error:**<br>STATUS0[MAM0_S3_A]<br>STATUS0[MAM0_S3_D]<br>STATUS0[MAM0_S3_AT]<br>STATUS0[MAM0_S3_M] | CH_13 | N/A |
| 19 | SCM Matrix Access Monitor Error Injection19 (DMA0) | 0~71 | 0~71 | 3~31 | 0~23 | **Master End Error:**<br>STATUS1[DMA0_A]<br>STATUS1[DMA0_D]<br>STATUS1[DMA0_M]<br>**MAM0_S3 Slave End Error:**<br>STATUS0[MAM0_S3_A]<br>STATUS0[MAM0_S3_D]<br>STATUS0[MAM0_S3_AT]<br>STATUS0[MAM0_S3_M] | CH_13 | N/A |
| 20 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 21 | SCM Matrix Access Monitor Error Injection21 (CPU0_AHBS) | 0~39 | 0~39 | 2~30 | N/A | **Slave End Error:**<br>STATUS2[C0_AHBS_A]<br>STATUS2[C0_AHBS_D]<br>STATUS2[C0_AHBS_M]<br>**CPU0/DMA0/HSM Master End Error:**<br>STATUS1[MAM0_S5_DS_A]<br>STATUS1[MAM0_S5_DS_D]<br>STATUS1[MAM0_S5_DS_M] | CPU0: CH_10<br>NONCPU: CH_13 | N/A |
| 22 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 23 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 24 | CPU0_ITCM | 0~71 | 0~71 | 3~23 | N/A | STATUS2[C0_ITCM_S]<br>STATUS2[C0_ITCM_M]<br>STATUS2[C0_ITCM_A] | CH_14 | CH_11 |

**Table 43. EIM matrix error check (bank channels) (continued)**

| EIM Bank Channels | | EIM BUS Register Field Configuration | | | | Error Check | | |
|---|---|---|---|---|---|---|---|---|
| CH. No. | Source Description | EIM_BUS_REG0[DATA0] Field Value | EIM_BUS_REG0[DATA1] Field Value | EIM_BUS_REG0[ADDR] Field Value | EIM_BUS_REG0[ATTR] Field Value | SCM Matrix Status Register | FCSMU CH. | ERM CH. |
| 25 | CPU0_DTCM0 | 0~39 | 0~39 | 2~23 | N/A | STATUS2[C0_DTCM0_S]<br>STATUS2[C0_DTCM0_M]<br>STATUS2[C0_DTCM0_A] | CH_14 | CH_12 |
| 26 | CPU0_DTCM1 | 0~39 | 0~39 | 2~23 | N/A | STATUS2[C0_DTCM1_S]<br>STATUS2[C0_DTCM1_M]<br>STATUS2[C0_DTCM1_A] | CH_14 | CH_13 |
| 27 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 28 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 29 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 30 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 31 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 32 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 33 | SRAM0_INJECTION | 0~71 | 0~71 | 3~31 | for sram0 control monitor:0~1 | STATUS6[SRAM0_S]<br>STATUS6[SRAM0_A]<br>STATUS6[SRAM0_EDC]<br>STATUS6[SRAM0_M]<br>STATUS6[SRAM0_CTRL_MON] | **For ECC error:** CH_6 **For EDC monitor:** CH_9 **For EDC error:** CH_28 | CH_8 |
| 34 | SRAM1_INJECTION | 0~71 | 0~71 | 3~31 | for sram1 control monitor:0~1 | STATUS6[SRAM1_S]<br>STATUS6[SRAM1_A]<br>STATUS6[SRAM1_EDC]<br>STATUS6[SRAM1_M]<br>STATUS6[SRAM1_CTRL_MON] | **For ECC error:** CH_6 **For EDC monitor:** CH_9 **For EDC error:** CH_28 | CH_9 |
| 35 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 36 | CPU0_DCACHE_DATA0_01 Injection | 0~38 for CPU0_DCACHE_DATA0_0 39~77 for CPU0_DCACHE_DATA0_1 | 0~38 for CPU0_DCACHE_DATA0_0 39~77 for CPU0_DCACHE_DATA0_1 | N/A | N/A | STATUS2[C0_DCACHE] | CH_14 | CH_15 |
| 37 | CPU0_DCACHE_DATA0_23 | 0~38 for CPU0_DCACH | 0~38 for CPU0_DCACH | N/A | N/A | STATUS2[C0_DCACHE] | CH_14 | CH_15 |

**Table 43. EIM matrix error check (bank channels) (continued)**

| EIM Bank Channels | | EIM BUS Register Field Configuration | | | | Error Check | | |
|---|---|---|---|---|---|---|---|---|
| CH. No. | Source Description | EIM_BUS_REG0[DATA0] Field Value | EIM_BUS_REG0[DATA1] Field Value | EIM_BUS_REG0[ADDR] Field Value | EIM_BUS_REG0[ATTR] Field Value | SCM Matrix Status Register | FCSMU CH. | ERM CH. |
| | Injection | E_DATA0_2 39~77 for CPU0_DCACHE_DATA0_3 | E_DATA0_2 39~77 for CPU0_DCACHE_DATA0_3 | | | | | |
| 38 | CPU0_DCACHE_DATA1_01 Injection | 0~38 for CPU0_DCACHE_DATA1_0 39~77 for CPU0_DCACHE_DATA1_1 | 0~38 for CPU0_DCACHE_DATA1_0 39~77 for CPU0_DCACHE_DATA1_1 | N/A | N/A | STATUS2[C0_DCACHE] | CH_14 | CH_15 |
| 39 | CPU0_DCACHE_DATA1_23 Injection | 0~38 for CPU0_DCACHE_DATA1_2 39~77 for CPU0_DCACHE_DATA1_3 | 0~38 for CPU0_DCACHE_DATA1_2 39~77 for CPU0_DCACHE_DATA1_3 | N/A | N/A | STATUS2[C0_DCACHE] | CH_14 | CH_15 |
| 40 | CPU0_DCACHE_TAG_01 Injection | 1~32 for CPU0_DCACHE_TAG_0 34~65 for CPU0_DCACHE_TAG_1 | 1~32 for CPU0_DCACHE_TAG_0 34~65 for CPU0_DCACHE_TAG_1 | N/A | N/A | STATUS2[C0_DCACHE] | CH_14 | CH_15 |
| 41 | CPU0_DCACHE_TAG_23 Injection | 1~32 for CPU0_DCACHE_TAG_2 34~65 for CPU0_DCACHE_TAG_3 | 1~32 for CPU0_DCACHE_TAG_2 34~65 for CPU0_DCACHE_TAG_3 | N/A | N/A | STATUS2[C0_DCACHE] | CH_14 | CH_15 |
| 42 | CPU0_ICACHE_DATA0 Injection | 0~71 for CPU0_ICACHE_DATA0 | 0~71 for CPU0_ICACHE_DATA0 | N/A | N/A | STATUS2[C0_ICACHE] | CH_14 | CH_14 |
| 43 | CPU0_ICACHE_DATA1 Injection | 0~71 for CPU0_ICACHE_DATA1 | 0~71 for CPU0_ICACHE_DATA1 | N/A | N/A | STATUS2[C0_ICACHE] | CH_14 | CH_14 |
| 44 | CPU0_ICACHE_TAG Injection | 2~20, 22~28 for CPU0_ICACHE_TAG0 31~49, 51~57 | 2~20, 22~28 for CPU0_ICACHE_TAG0 31~49, 51~57 | N/A | N/A | STATUS2[C0_ICACHE] | CH_14 | CH_14 |

**Table 43. EIM matrix error check (bank channels) (continued)**

| EIM Bank Channels | | EIM BUS Register Field Configuration | | | | Error Check | | |
|---|---|---|---|---|---|---|---|---|
| CH. No. | Source Description | EIM_BUS_REG0[DATA0] Field Value | EIM_BUS_REG0[DATA1] Field Value | EIM_BUS_REG0[ADDR] Field Value | EIM_BUS_REG0[ATTR] Field Value | SCM Matrix Status Register | FCSMU CH. | ERM CH. |
| | | for CPU0_ICACHE_TAG1 | for CPU0_ICACHE_TAG1 | | | | | |
| 45 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 46 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 47 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 48 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 49 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 50 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 51 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 52 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 53 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 54 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 55 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 56 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 57 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 58 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 59 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 60 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 61 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 62 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 63 | SCM S5 64:32 DownSize Monitor Error Injection | for downsize monitor: 0~1 | for downsize monitor: 0~1 | N/A | N/A | STATUS6[MAM0_S5_DS] | CH_13 | N/A |
| 64 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 65 | DMA0_CFG_ECC Injection | for data: 0~63 for data chkbit: 64~103 for address chkbit: 104~108 | for data: 0~63 for data chkbit: 64~103 for address chkbit: 104~108 | 0~5 | N/A | STATUS5[DMA0_CFG_S] STATUS5[DMA0_CFG_M] STATUS5[DMA0_CFG_A] | CH_23 | CH_4 |
| 66 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 67 | ROM_ECC Injection | 0~71 | 0~71 | 3~31 | N/A | STATUS0[ROM_A] STATUS0ROM_D] STATUS0[ROM_M] | CPU0: CH_10 NONCPU: CH_13 | CH_6 |

**Table 43. EIM matrix error check (bank channels) (continued)**

| EIM Bank Channels | | EIM BUS Register Field Configuration | | | | Error Check | | |
|---|---|---|---|---|---|---|---|---|
| CH. No. | Source Description | EIM_BUS_REG0[DATA0] Field Value | EIM_BUS_REG0[DATA1] Field Value | EIM_BUS_REG0[ADDR] Field Value | EIM_BUS_REG0[ATTR] Field Value | SCM Matrix Status Register | FCSMU CH. | ERM CH. |
| 68 | SCM S5 Downsize ECC Check with AHBS Error Injection | 0~39 | 0~39 | 2~30 | N/A | CPU0/DMA0HSM Master End Error:<br>STATUS1[MAM0_S5_DS_A]<br>STATUS1[MAM0_S5_DS_D]<br>STATUS1[MAM0_S5_DS_M]<br>CPU0 AHBS Slave End Error:<br>STATUS2[C0_AHBS_A]<br>STATUS2[C0_AHBS_D]<br>STATUS2[C0_AHBS_M] | CPU0:<br>CH_10<br>NONCPU:<br>CH_13 | N/A |
| 69 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 70 | SCM CPU0 AHBP and AHBS Monitor Error Injection | 0~1 For CPU0 AHBP Monitor<br>64~65 For AHBS Monitor | 0~1 For CPU0 AHBP Monitor<br>64~65 For AHBS Monitor | N/A | N/A | AHBP End:<br>STATUS2[C0_AHBP_F2S]<br>AHBS End:<br>STATUS6[MAM0_S5_S2F] | AHBP:<br>CH_10<br>AHBS:<br>CH_13 | N/A |
| 71 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 72 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 73 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 74 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 75 | SCM S8 64:32 DownSizeMonitor Error Injection | for downsize monitor: 0~1 | for downsize monitor: 0~1 | N/A | N/A | STATUS6[MAM0_S8_DS] | CH_13 | N/A |
| 76 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 77 | RAM Decoder Monitor Error Injection | 0~1 for CPU0 ITCM Decoder Monitor<br>4~7 for CPU0 DTCM0 Decoder Monitor<br>8~11 for CPU0 DTCM1 Decoder Monitor | 0~1 for CPU0 ITCM Decoder Monitor<br>4~7 for CPU0 DTCM0 Decoder Monitor<br>8~11 for CPU0 DTCM1 Decoder Monitor | 0~3 for System RAM0 Decoder Monitor<br>8~11 for System RAM1 Decoder Monitor | 0~3 for System RAM0 Decoder Monitor<br>8~11 for System RAM1 Decoder Monitor | STATUS6[SRAM0_DEC]<br>STATUS6[SRAM1_DEC]<br>STATUS2[C0_ITCM_DEC]<br>STATUS2[C0_DTCM0_DEC]<br>STATUS2[C0_DTCM1_DEC] | **CPU0 TCM Decoder Monitor Error:**<br>CH_14<br>**SRAM0/1 Decoder Monitor Error:**<br>CH_9 | N/A |
| 78 | AFCB0 Monitor Error | 0~1 | 0~1 | N/A | N/A | STATUS6[AFCB0_MON] | CH_28 | N/A |

**Table 43. EIM matrix error check (bank channels) (continued)**

| EIM Bank Channels | | EIM BUS Register Field Configuration | | | | Error Check | | |
|---|---|---|---|---|---|---|---|---|
| CH. No. | Source Description | EIM_BUS_REG0[DATA0] Field Value | EIM_BUS_REG0[DATA1] Field Value | EIM_BUS_REG0[ADDR] Field Value | EIM_BUS_REG0[ATTR] Field Value | SCM Matrix Status Register | FCSMU CH. | ERM CH. |
| | Injection | | | | | | | |
| 79 | AFCB1 Monitor Error Injection | 0~1 | 0~1 | N/A | N/A | STATUS6[AFCB1_MON] | CH_28 | N/A |
| 80 | CPU0 Overlay Error Injection | 0~1 | 0~1 | N/A | N/A | STATUS2[C0_OVERLAY] | CH_4 | N/A |
| 81 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 82 | HSM DRAM ECC Error Injection | 0~39 | 0~39 | 2~31 | N/A | STATUS7[HSM_DRAM_S] STATUS7[HSM_DRAM_M] | CH_28 | CH_26 |
| 83 | HSM IRAM ECC Error Injection | 0~39 | 0~39 | 2~31 | N/A | STATUS7[HSM_IRAM_S] STATUS7[HSM_IRAM_M] | CH_28 | CH_27 |

***Note:*** For EIM Bank Channels 11, 12, 18, and 19, the complete bit fields of slave end errors are as follows:

- **For Slave MAM0_S0 End Error:** N/A
- **For Slave MAM0_S1 End Error:** SCM_MATRIX_STATUS0[MAM0_S1_A], SCM_MATRIX_STATUS0[MAM0_S1_D], SCM_MATRIX_STATUS0[MAM0_S1_AT], SCM_MATRIX_STATUS0[MAM0_S1_M]
- **For Slave MAM0_S2 End Error:** SCM_MATRIX_STATUS0[MAM0_S2_A], SCM_MATRIX_STATUS0[MAM0_S2_D], SCM_MATRIX_STATUS0[MAM0_S2_AT], SCM_MATRIX_STATUS0[MAM0_S2_M]
- **For Slave MAM0_S3 End Error:** SCM_MATRIX_STATUS0[MAM0_S3_A], SCM_MATRIX_STATUS0[MAM0_S3_D], SCM_MATRIX_STATUS0[MAM0_S3_AT], SCM_MATRIX_STATUS0[MAM0_S3_M]
- **For Slave MAM0_S4 End Error:** SCM_MATRIX_STATUS0[MAM0_S4_A], SCM_MATRIX_STATUS0[MAM0_S4_D], SCM_MATRIX_STATUS0[MAM0_S4_AT], SCM_MATRIX_STATUS0[MAM0_S4_M]
- **For Slave MAM0_S5 End Error:** SCM_MATRIX_STATUS0[MAM0_S5_A], SCM_MATRIX_STATUS0[MAM0_S5_D], SCM_MATRIX_STATUS0[MAM0_S5_AT], SCM_MATRIX_STATUS0[MAM0_S5_M]
- **For Slave MAM0_S6 End Error:** SCM_MATRIX_STATUS1[MAM0_S6_A], SCM_MATRIX_STATUS1[MAM0_S6_D], SCM_MATRIX_STATUS1[MAM0_S6_AT], SCM_MATRIX_STATUS1[MAM0_S6_M]
- **For Slave MAM0_S7 End Error:** SCM_MATRIX_STATUS1[MAM0_S7_A], SCM_MATRIX_STATUS1[MAM0_S7_D], SCM_MATRIX_STATUS1[MAM0_S7_AT], SCM_MATRIX_STATUS1[MAM0_S7_M]
- **For Slave MAM0_S8 End Error:** SCM_MATRIX_STATUS1[MAM0_S8_A], SCM_MATRIX_STATUS1[MAM0_S8_D], SCM_MATRIX_STATUS1[MAM0_S8_AT], SCM_MATRIX_STATUS1[MAM0_S8_M]

**Table 44. EIM matrix error check (lockstep channels)**

| EIM Lockstep Channels | EIM LOCKSTEP Register | | | | SCM Matrix Status Register | FCSMU CH. | ERM CH. |
|---|---|---|---|---|---|---|---|
| | LOCKSTEP_ | LOCKSTEP_ | LOCKSTEP | LOCKSTEP | | | |

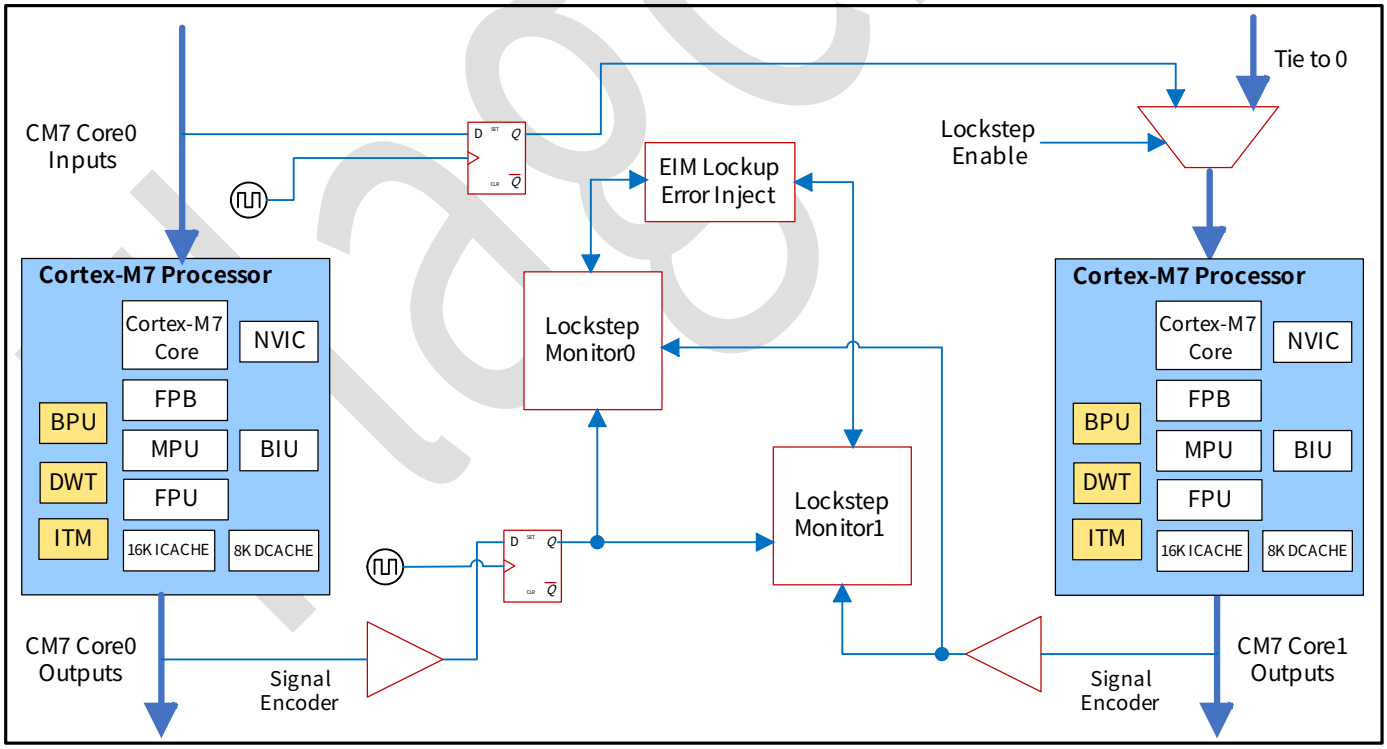
| | | MON0_SET Field | MON1_SET Field | _MON0_CLR Field | _MON1_CLR Field | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | CPU0_LOCKSTEP Injection | 1 For CPU0 Lockstep MON0 Error Injection | 1 For CPU0 Lockstep MON1 Error Injection | 1 For CPU0 Lockstep MON0 Error Clean | 1 For CPU0 Lockstep MON1 Error Clean | STATUS2[C0_LOCKSTEP] | CH_4 | N/A |
| 1 | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| 2 | DMA0_LOCKSTEP Injection | 1 For DMA0 Lockstep MON0 Error Injection | 1 For DMA0 Lockstep MON1 Error Injection | 1 For DMA0 Lockstep MON0 Error Clean | 1 For DMA0 Lockstep MON1 Error Clean | STATUS5[DMA0_LOCKSTEP] | CH_23 | N/A |

### 14.2.4 Lockstep Control

The SCM implements the lockstep enable control, and the lockup enable is not possible to be changed by the user. The EIM implements the error injection function for the lockstep monitors. The lockstep monitors implement two lockstep monitor logics to avoid a latent failure of the lockstep monitor logics. Lockstep monitor errors are sent to the FCSMU for fault handling. The CPU lockstep monitors cover the ICACHE ECC, DCACHE ECC, ITCM ECC, DTCM ECC, and AXI2AHB interface, but do not cover the cache array, TCM array, CTI, etc.

The DMA lockstep implements the DMA monitor, which only includes the DMA engine, and the DMA mux is not included in the lock step monitor.

**Figure 29. Lockstep diagram**




### 14.2.5 INTM

- Support 4-channels interrupt monitoring. All the CM7 NVIC interrupts can be routed to the INTM
- Support active interrupt or in-active interrupt timeout monitor
- For interrupt monitoring, the INTM can be configured to generate an interrupt or reset if an error occurs to the monitored interrupt over a certain period

### 14.2.6 CMU

- Support checking the frequency out of the preconfigured range
- Support the loss of clock detection
- Support the Period Monitor mode triggered by a hardware trigger, with the trigger source from internal
- Support CMU0-3 to monitor system clock sources including SIRC32K/FOSC/FIRC96/SIRC12M. SCG provides the PLL0/PLL1 and system slow clock monitor. CMU4 also uses FOSC or SIRC12 to monitor the slow clock

### 14.2.7 WDOG

- Support two WDOGs
- Support the window refresh capability
- Support independent clocks from the system clock source
- Support register lock for the critical configuration register

### 14.2.8 ISM

- Support 16 FPC input channels, through TRGSEL3
- Support 16 LAM channels as monitor sources
- Support the input glitch filter function
- Support delay monitor between the reference source and monitor source

### 14.2.9 TMU

- Support +/-3°C accuracy at hot and +/-5°C over temperature
- Temperature sensor output to the ADC
- Temperature monitor comparator to generate an alarm at two temperature conditions

### 14.2.10 CRC

- Support the 8-bit/16-bit/32-bit poly
- Support the 8-bit/16-bit/32-bit data operation
- Support the data swap function

Flagchip

# Chapter 15 Pinouts and Packaging

## 15.1 Pinouts

For the FC7240 family, except for the pins listed in the below table *Reset status of pins*, the reset status of PAD control bits (PORT_PCR[7:0]) of the rest pins is "Disable." In addition, only pull-up instead of pull-down is available to the input-only pads (PTE3/14).

**Table 45. Reset status of pins**

| Pad | Reset Status |
|---|---|
| PTA4 | PULL enable and PULL up |
| PTA11 | PFE enable |
| PTC4 | PULL enable and PULL down |
| PTC5 | PULL enable and PULL up |
| PTE3/14 | PULL disable but PULL up |
| PTD2 | PULL disable but PULL up |

The following table lists the ports which belong to the PD0, and the ports not mentioned here belong to the PD1.

**Table 46. Ports belonging to PD0**

| Port | Pin Number |
|---|---|
| A | 4, 8, 9, 10, 11, 12, 13, 14, 18, 19, 20, 21 |
| C | 4, 5 |
| D | 0, 1 |
| E | 0, 1, 10, 11, 15, 16, 19, 20, 21, 22, 23, 24, 25, 26, 27, 29 |

For FC7240F2M, the NMI function works normally and is disabled by default. To enable the NMI function, program Bit 1 of BOOT Configuration 0 (BC0) to 0. In this condition, if the NMI function is not desired, it is recommended to select the GPIO function of PTA11.

The pads of the FC7240 family are divided into normal pads, input-only pads, and High-Speed (HS) pads. The table below list the pads that belong to input-only and HS.

**Table 47. Pad type**

| Pad Type | Pad |
|---|---|
| Input-only | PTE14, PTE3 |
| HS | PTE8, PTB5, PTB4, PTC3, PTC2, PTD7, PTD6, PTD5, PTD12, PTD11, PTD10, PTC0, PTB22 |

For the detailed pin mux mapping, refer to the attached *FC7240 Pinout.xlsx*.

## 15.2 Package

There are three packages available for the FC7240F2M: LQFP176-EP, LQFP144-EP, and LQFP100-EP.

For detailed package information, refer to *FC7240 Data Sheet*.

# Chapter 16 CPU Private Module (CPM)

## 16.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • Added a new bit field "DIS_AHBD_MON" to the register section *CPM_MISCR;*<br>• In the register section *CPM_TCMRCR:*<br>- Changed the access attribute for the bit fields "D1RBIE", "D0RBIE", "IRBIE", "D1RBAC", "D0RBAC", and "IRBAC" from "RO" to "RW";<br>- Updated descriptions of the bit fields "D1RBIE", "D0RBIE", and "IRBIE" |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 16.2 Features

The CPU Private Module (CPM) includes the following features:

- FPU exception interrupt control and status retrieval
- Determination of arbitration priority between the software and backdoor access
- TCM Retry control and status

## 16.3 Overview

The CPM provides miscellaneous control functions and status information about FPU exceptions, and determines priority between software and backdoor access.

The CPM provides the control function and status information about TCM retry functionality.

## 16.4 Functional Description

This section describes the functional description of the CPM module.

### 16.4.1 FPU Exception Report

An CPM interrupt will be generated if any of the following is true:

- FPU inexact interrupt is enabled (FIXCE) and a number is inexact (FIXC)
- FPU underflow interrupt is enabled (FUFCE) and an underflow occurs (FUFC)
- FPU overflow interrupt is enabled (FOFCE) and an overflow occurs (FOFC)
- FPU divide-by-zero interrupt is enabled (FDZCE) and a divide-by-zero occurs (FDZC)
- FPU invalid operation interrupt is enabled (FIOCE) and an invalid occurs (FIOC)

If an interrupt status field and its enabling field in the FISCR register are all set, an interrupt will be driven by it. This feature is designed to report FPU exceptions to the software application immediately.

### 16.4.2 Access Priority Control

The ITCM and DTCM under per core can not only be accessed by the core it belongs to, but also by other masters via the backdoor bus interface AHBS. The CPM supports priority control between software access and AHBS access.

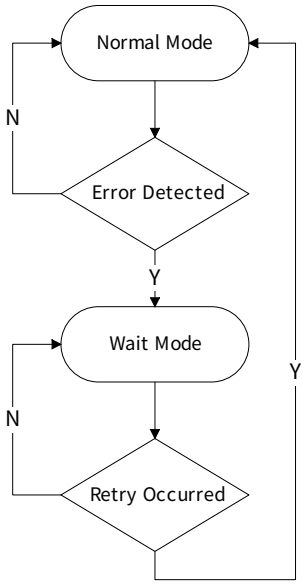
### 16.4.3 TCM Retry Control

In default, TCM (including ITCM, DTCM0, and DTCM1) works under normal mode, with no wait inserted for each TCM read. This mode has a better performance by checking ECC after TCM read.

Once an ECC error is detected in normal mode, TCM will change to work under wait mode after issuing a TCM Retry flag to the core. In wait mode, TCM will insert one wait cycle for each TCM read, and check ECC in the wait cycle before feeding back read data to core. This mode has worse performance since wait cycles are inserted.

When entering wait mode, a retry flag is sent to the core, and the address of ECC failure is latched in the TCM retry buffer. Each TCM read will be checked with the TCM retry buffer after the TCM retry buffer is valid. Once a match is found between the address of the TCM read and the TCM retry buffer, TCM will be switched to normal mode and clear the TCM retry buffer.

Flagchip




The condition to switch TCM from normal mode to wait mode is ECC error is detected.

The condition to switch TCM from wait mode to normal mode is any of the following:

- TCM retry operation is performed by core (address of TCM read matches TCM retry buffer).
- After 64 TCM reads when TCM auto clear (D1RBAC/D0RBAC/IRBAC) is set in CPM.
- Write 1 to TCM retry buffer force clear bit, to force exit wait mode.

Also provides the capability of interrupt for time out on waiting retry. When set D1RBIE/D0RBIE/IRBIE, and the TCM retry buffer is valid (an ECC error is detected), an interrupt will be generated after 64 TCM reads without TCM retry performed.

## 16.5 Register Definition

CPM base address : E008_0000h

**Register Overview - CPM**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | CPM_FISCR (FPU Interrupt Status and Control Register) | 32 | RW | 0000_0000h | page |
| **4h** | CPM_MISCR (Miscellaneous Control Register) | 32 | RW | 0000_0100h | page |
| **20h** | CPM_TCMRCR (TCM Retry Control Register) | 32 | RW | 0000_0000h | page |

### 16.5.1 CPM_FISCR (FPU Interrupt Status and Control Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | FIXCE | FIDCE | FOFCE | FUFCE | FDZCE | FIOCE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | FIXC | FIDC | FOFC | FUFC | FDZC | FIOC |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CPM_FISCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **21**<br>**FIXCE** | RW | FPU Inexact Interrupt Enable<br>0b - Disable interrupt<br>1b - Enable interrupt |
| **20**<br>**FIDCE** | RW | FPU Input Denormal Interrupt Enable<br>0b - Disable interrupt<br>1b - Enable interrupt |
| **19**<br>**FOFCE** | RW | FPU Overflow Interrupt Enable<br>0b - Disable interrupt<br>1b - Enable interrupt |
| **18**<br>**FUFCE** | RW | FPU Underflow Interrupt Enable<br>0b - Disable interrupt<br>1b - Enable interrupt |
| **17**<br>**FDZCE** | RW | FPU Divide-by-Zero Interrupt Enable<br>0b - Disable interrupt<br>1b - Enable interrupt |
| **16**<br>**FIOCE** | RW | FPU Invalid Operation Interrupt Enable<br>0b - Disable interrupt<br>1b - Enable interrupt |
| **5**<br>**FIXC** | RO | FPU Inexact Interrupt Status<br>This field is a copy of the core's FPSCR[IXC] field and signals an inexact number has been detected in the processor's FPU. Once set, this field remains set until software clears FPSCR[IXC].<br>0b - No interrupt<br>1b - Interrupt occurred |
| **4**<br>**FIDC** | RO | FPU Input Denormal Interrupt Status<br>This field is a copy of the core's FPSCR[IDC] field and signals input denormalized number has been detected in the processor's FPU. After the field is set, it remains set until software clears FPSCR[IDC].<br>0b - No interrupt<br>1b - Interrupt occurred |

| | | |
|---|---|---|
| **3**<br>**FOFC** | RO | FPU Overflow Interrupt Status<br>This field is a copy of the core's FPSCR[OFC] field and signals an overflow has been detected in the processor's FPU. After this field is set, it remains set until software clears FPSCR[OFC].<br>0b - No interrupt<br>1b - Interrupt occurred |
| **2**<br>**FUFC** | RO | FPU Underflow Interrupt Status<br>This field is a copy of the core's FPSCR[UFC] field and signals an underflow has been detected in the processor's FPU. After this field is set, it remains set until software clears FPSCR[UFC].<br>0b - No interrupt<br>1b - Interrupt occurred |
| **1**<br>**FDZC** | RO | FPU Divide-by-Zero Interrupt Status<br>This field is a copy of the core's FPSCR[DZC] field and signals a divide by zero has been detected in the processor's FPU. After this field is set, it remains set until software clears FPSCR[DZC].<br>0b - No interrupt<br>1b - Interrupt occurred |
| **0**<br>**FIOC** | RO | FPU Invalid Operation Interrupt Status<br>This field is a copy of the core's FPSCR[IOC] field and signals an illegal operation has been detected in the processor's FPU. After this field is set, it remains set until software clears FPSCR[IOC].<br>0b - No interrupt<br>1b - Interrupt occurred |

### 16.5.2 CPM_MISCR (Miscellaneous Control Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | DIS_AHBD_MON | 0 | | | | | | | AHBS_PRIORITY |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CPM_MISCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **8**<br>**DIS_AHBD_MON** | RW | Disable AHBD Monitor<br>0b - Enble AHBD monitor<br>1b - Disable AHBD monitor |
| **0**<br>**AHBS_PRIORITY** | RW | AHBS Priority Control<br>0b - A software access takes priority over an AHBS access<br>1b - An AHBS access has priority over a software access |

### 16.5.3 CPM_TCMRCR (TCM Retry Control Register)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | D1RBCLR | D0RBCLR | IRBCLR | 0 | D1RBVLD | D0RBVLD | IRBVLD | 0 | D1RBIE | D0RBIE | IRBIE | 0 | D1RBAC | D0RBAC | IRBAC |
| W | | | | | | wc | wc | wc | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CPM_TCMRCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **14**<br>**D1RBCLR** | RW | DTCM1 Retry Buffer Force Clear<br>0b - No operation<br>1b - Force clear DTCM1 retry buffer |
| **13**<br>**D0RBCLR** | RW | DTCM0 Retry Buffer Force Clear<br>0b - No operation<br>1b - Force clear DTCM0 retry buffer |
| **12**<br>**IRBCLR** | RW | ITCM Retry Buffer Force Clear<br>0b - No operation<br>1b - Force clear ITCM retry buffer |

| | | |
|---|---|---|
| **10**<br>**D1RBVLD** | WC | DTCM1 Retry Buffer Valid<br>0b - Retry buffer is invalid, no retry is issued.<br>1b - Retry buffer is valid. DTCM1 retry is issued, and not cleared yet. One cycle RAM wait is inserted for each DTCM1 read.<br>Note: Write 0 or 1 to clear (WC). |
| **9**<br>**D0RBVLD** | WC | DTCM0 Retry Buffer Valid<br>0b - Retry buffer is invalid, no retry is issued.<br>1b - Retry buffer is valid. DTCM0 retry is issued, and not cleared yet. One cycle RAM wait is inserted for each DTCM0 read.<br>Note: Write 0 or 1 to clear (WC). |
| **8**<br>**IRBVLD** | WC | ITCM Retry Buffer Valid<br>0b - Retry buffer is invalid, no retry is issued.<br>1b - Retry buffer is valid. ITCM retry is issued, and not cleared yet. One cycle RAM wait is inserted for each ITCM read.<br>Note: Write 0 or 1 to clear (WC). |
| **6**<br>**D1RBIE** | RW | DTCM1 Retry Buffer Timeout Interrupt Enable<br>This field enables DTCM1 Retry buffer timeout interrupt. When this bit is set, an interrupt will be generated after 64 DTCM1 reads without DTCM1 retry operation happened when DTCM1 Retry buffer is valid.<br>0b - No interrupt<br>1b - Interrupt enable |
| **5**<br>**D0RBIE** | RW | DTCM0 Retry Buffer Timeout Interrupt Enable<br>This field enables DTCM0 Retry buffer timeout interrupt. When this bit is set, an interrupt will be generated after 64 DTCM0 reads without DTCM0 retry operation happened when DTCM0 Retry buffer is valid.<br>0b - No interrupt<br>1b - Interrupt enable |
| **4**<br>**IRBIE** | RW | ITCM Retry Buffer Timeout Interrupt Enable<br>This field enables ITCM Retry buffer timeout interrupt. When this bit is set, an interrupt will be generated after 64 ITCM reads without ITCM retry operation happened when ITCM Retry buffer is valid.<br>0b - No interrupt<br>1b - Interrupt enable |
| **2**<br>**D1RBAC** | RW | DTCM1 Retry Buffer Autoclear<br>This field enables auto clear functionality for DTCM1 retry buffer. When this bit is set, DTCM1 retry buffer will be set to invalid after 64 DTCM1 reads if no DTCM1 retry operation happened.<br>0b - No operation<br>1b - Enable auto clear for DTCM1 retry buffer |

| | | |
|---|---|---|
| **1**<br>**D0RBAC** | RW | DTCM0 Retry Buffer Autoclear<br>This field enables auto clear functionality for DTCM0 retry buffer. When this bit is set, DTCM0 retry buffer will be set to invalid after 64 DTCM0 reads if no DTCM0 retry operation happened.<br>0b - No operation<br>1b - Enable auto clear for DTCM0 retry buffer |
| **0**<br>**IRBAC** | RW | ITCM Retry Buffer Autoclear<br>This field enables auto clear functionality for ITCM retry buffer. When this bit is set, ITCM retry buffer will be set to invalid after 64 ITCM reads if no ITCM retry operation happened.<br>0b - No operation<br>1b - Enable auto clear for ITCM retry buffer |

Flagchip

# Chapter 17 Direct Memory Access (DMA)

## 17.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • Added general descriptions to the register sections *DMA_CFG_NBYTES_ILNOn, DMA_CFG_NBYTES_ILOFFNOn, DMA_CFG_NBYTES_ILOFFYESn, DMA_CFG_CLC_CHTRGENNOn, DMA_CFG_CLC_CHTRGENYESn, DMA_CFG_BLC_CHTRGENNOn,* and *DMA_CFG_BLC_CHTRGENYESn;*<br>• Changed the address for the register "DMAMUX_CHTRG" from "20h" to "10h" in Section *DMAMUX Register Definition* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 17.2 Chip-specific Information

N/A

## 17.3 Features

- All data movement operations are all read from the source address and written to the destination address with programmable source and destination addresses and transfer size
- Support different addressing modes and support source or destination address un-align modulo
- Support nested transfer operations by programmable transfer configuration (CFG)
- 32-byte transfer CFG stored in the DMA local memory for each channel
- An inner data transfer loop defined by an inner loop count
- An outer data transfer loop defined by an outer loop count
- Support three methods to active a DMA channel:
  - Internal software start
  - Peripheral hardware requests
  - Initiation via a channel-to-channel trigger mechanism for continuous transfers
- Support channel fixed-priority and round-robin channel arbitration
- Support interrupt request assertion to report channel completion
  - One interrupt per channel, which asserted at the completion of outer loop count or half of outer loop count
- Programmable error terminations per channel and logically OR together to form one error interrupt assert
- Support the additionally periodic trigger modes for the first 4 channels
- Support up to 125 peripheral slots and up to 2 always-on slots can be routed to 16 channels with DMAMUX
- Support 16 independently selectable DMA channel routers
- Each channel router can be assigned to one of the peripheral DMA slots or to one of the always-on slots.
- Support DMA local memory Error Correction Code (ECC) function, such as address and data ECC:
  - Inject errors to DMA local memory with EIM channel
  - Programmable single-bit error correction interrupt reporting with ERM channel
  - Programmable multi bits error no-correction interrupt reporting with ERM channel
  - Record the faulting offset address (align) the last ECC event on DMA local memory with the ERM channel
- Support DMA lockstep monitor.

## 17.4 Overview

The Direct Memory Access (DMA) controller module, as a highly programmable data-transfer engine, has the capacity to execute complex data transfers with minimal intervention from a host processor. It can be used to transfer data flexibly. The main hardware microarchitecture includes three parts:

- A DMA engine that performs source address and destination address calculations, and data movement operation from a source address to a destination address.
- Local memory containing transfer configurations for each of the 16 channels. The local memory must be initialized before being accessed. The RAM range starts from CFG_SADDRn to CFG_BLC_CHTRGENYESn (N: 0 ~ 15).
- The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources to any of the 16 DMA channels.
- The DMA lockstep implements the DMA monitor which only includes the DMA engine, the DMAMUX is not included in the lockstep monitor.

***Note:*** To perform the software reset, take the following four steps:

1) Set the DMA PCC software reset.

2) Disable the DMA PCC clock.
3) Clear the DMA PCC software reset.
4) Enable the DMA PCC clock.

## 17.5 Functional Description

### 17.5.1 DMA Initialization

To initialize the DMA channel request service, perform the following steps:

1) Program the CR register with the desired configuration.
2) Program the DCHPRIn registers with the desired channel priority levels.
3) Program the EEI register to optionally enable error interrupts.
4) Program the 32-byte CFG for each channel that may request a service.
5) Program the ERQ register to enable any hardware service requests.
6) Request a channel service via either setting the CFG_CSRn[START] or peripheral assert DMA request.

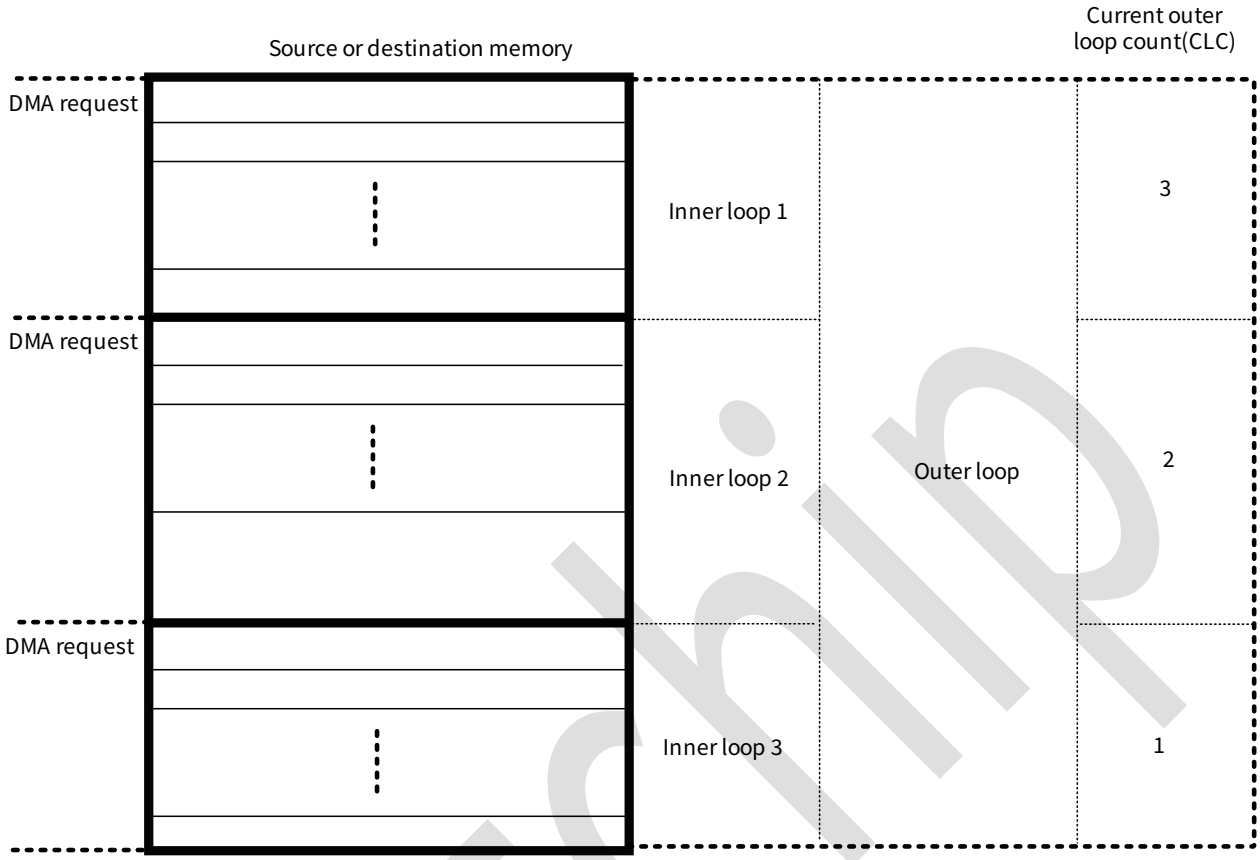
A channel is performed data transfer according to the channel arbitration and channel priority levels written into the programmer's model. The DMA engine reads the entire CFG from the programmer's model into its internal address path module for the selected channel. Initiating the first transfer on the internal bus as the CFG, unless a channel configuration error is detected. Transfers from the source, defined by CFG_SADDRn, to the destination, defined by CFG_DADDRn, continue until the number of bytes defined by CFG_NBYTESn are transferred.

When the transfer is completed, the DMA engine's local CFG_SADDRn, CFG_DADDRn, and CFG_CLCn are written back to the main CFG memory and any inner loop channel trigger is performed, if enabled. If the outer loop is done, further post-processing executes, such as interrupts, and outer loop channel trigger, if enabled.

The figure below explains the process that one inner-loop or loop transfer is enabled by the DMA request without CPU intervention. DMA arbitration occurs after each inner loop and can preempt one level of the inner loop. The beginning loop count (BLC) sets the number of inner loops in an outer loop.

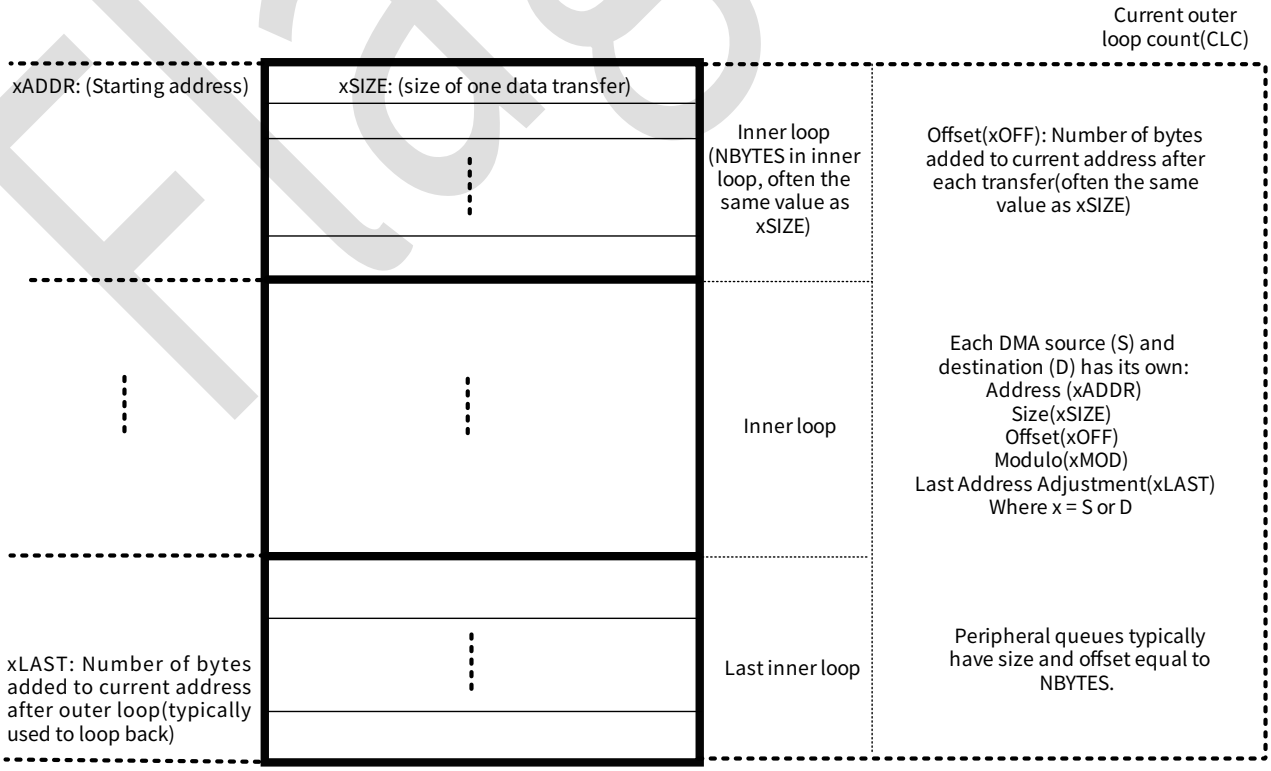
Flagchip

**Figure 30. Example of nested loop**




The following figure shows the memory array terms and the CFG settings interrelation.

**Figure 31. Memory array terms**




Flagchip

### 17.5.2 Channel Programming Errors

The DMA executes all kinds of tests on the Transfer configuration to verify if the descriptor data is consistent. Most programming errors are reported on a per-channel basis except for a channel priority error (ES[CPE]).

The error channel number caused by error types except channel priority error is recorded in the Error Status register (DMA ES). The error source will be detected and reported again if the error source still exists before activating the problem channel next time.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest channel priority with an active request is selected, but the lowest numbered channel with that priority is selected by arbitration and executed by the DMA engine. The hardware service request handshake signals, error interrupts, and error reporting are associated with the selected channel.

### 17.5.3 Channel Arbitration Mode

In fixed channel arbitration mode, choose the highest priority channel service to execute the request service.

In round-robin channel arbitration mode, a channel services from the highest channel number to the lowest, without considering the channel priority levels.

### 17.5.4 DMA Transfer Operations

#### *17.5.4.1 Single Request*

For a single request, set the outer loop count to one (CFG_CLCn=CFG_BLCn=1) to perform a simple transfer of n bytes data with one activation. The data transfer starts after the channel service request is acknowledged and the channel is selected to execute. After completing the transfer, the CFG_CSRn[DONE] bit is set and the channel complement transfer interrupt request asserts if enabled.

For example, the following CFG entry is configured to transfer 32 bytes of data. The DMA is programmed for one outer loop transferring 32 bytes per loop. The source memory has a byte-wide memory port and the start address is 0x2000. The destination memory has a 32-bit wide memory port and the start address is 0x4000. The address offsets are programmed in increments to match the transfer size: one byte for the source address and four bytes for the destination address. The last source and destination addresses are adjusted to return to their beginning values.

- CFG_CLCn = CFG_BLCn = 1　CFG_NBYTESn = 32
- CFG_SADDRn = 0x2000　CFG_SOFFn = 1
- CFG_ATTRn[SSIZE] = 0　CFG_SLASTn = -32
- CFG_DADDRn = 0x4000　CFG_DOFFn = 4
- CFG_ATTRn[DSIZE] = 2　CFG_DLASTn= –32
- CFG_CSRn[INTOUTER] = 1　CFG_CSRn[START] = 1

Based on the above CFG: generates the following event sequence:

- Program the CFG_CSRn[START] bit requests channel service.
- The channel is selected by arbitration for servicing.
- DMA engine writes: CFG_CSRn[DONE] = 0, CFG_CSRn[START] = 0, CFG_CSRn[ACTIVE] = 1.
- DMA engine reads: channel CFGn data from local memory to the internal register file.
- The source-to-destination transfers are executed as follows steps:
  - Read byte from location 0x2000, read byte from location 0x2001, read byte from 0x2002, read byte from 0x2003. Write 32-bits to location 0x4000, the first loop of the inner loop.
  - Read byte from location 0x2004, read byte from location 0x2005, read byte from 0x2006, read byte from 0x2007.

Write 32-bits to location 0x4004, the second loop of the inner loop.

- Read byte from location 0x2008, read byte from location 0x2009, read byte from 0x200A, read byte from 0x200B. Write 32-bits to location 0x4008, the third loop of the inner loop.
- Read byte from location 0x200C, read byte from location 0x200D, read byte from 0x200E, read byte from 0x200F. Write 32-bits to location 0x400C, the fourth loop of the inner loop.
- Read byte from location 0x2010, read byte from location 0x2011, read byte from 0x2012, read byte from 0x2013. Write 32-bits to location 0x4010, the fifth loop of the inner loop.
- Read byte from location 0x2014, read byte from location 0x2015, read byte from 0x2016, read byte from 0x2017. Write 32-bits to location 0x4014, the sixth loop of the inner loop.
- Read byte from location 0x2018, read byte from location 0x2019, read byte from 0x201A, read byte from 0x201B. Write 32-bits to location 0x4018, the seventh loop of the inner loop.
- Read byte from location 0x201C, read byte from location 0x201D, read byte from 0x201E, read byte from 0x201F. Write 32-bits to location 0x401C, the last loop of the inner loop, and the one outer loop complete.

- The DMA engine writes: CFG_SADDRn = 0x2000, CFG_DADDRn = 0x4000, CFG_CLCn = 1 (CFG_BLCn).
- The DMA engine writes: CFG_CSRn[ACTIVE] = 0, CFG_CSRn[DONE] = 1, INT[n] = 1.
- The channel retires and the DMA goes idle or services the next channel.

#### *17.5.4.2 Multiple requests*

The following example transfers 64 bytes via two hardware requests, but the difference between this and the previous example is that the outer loop count and the last address offsets. The DMA is programmed for two loops of the outer loop transferring 32 bytes per loop. After the channel's hardware requests are enabled in the ERQ register, the peripheral initiates channel service requests.

- CFG_CLCn = CFG_BLCn = 2
- CFG_SLASTn = –64
- CFG_DLASTn = –64

Based on the above CFG: generates the following event sequence:

- First hardware request: the peripheral asserts the DMA request for channel service.
- The channel is selected by arbitration for servicing.
- DMA engine writes: CFG_CSRn[DONE] = 0, CFG_CSRn[START] = 0, CFG_CSRn[ACTIVE] = 1.
- DMA engine reads: channel CFGn data from local memory to the internal register file.
- The source to destination transfers are executed in the following steps:
  - Read byte from location 0x2000, read byte from location 0x2001, read byte from 0x2002, read byte from 0x2003. Write 32-bits to location 0x4000, the first loop of the inner loop.
  - Read byte from location 0x2004, read byte from location 0x2005, read byte from 0x2006, read byte from 0x2007. Write 32-bits to location 0x4004, the second loop of the inner loop.
  - Read byte from location 0x2008, read byte from location 0x2009, read byte from 0x200A, read byte from 0x200B. Write 32-bits to location 0x4008, the third loop of the inner loop.
  - Read byte from location 0x200C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F. Write 32-bits to location 0x400C, the fourth loop of the inner loop.
  - Read byte from location 0x2010, read byte from location 0x2011, read byte from 0x2012, read byte from 0x2013. Write 32-bits to location 0x4010, the fifth loop of the inner loop.
  - Read byte from location 0x2014, read byte from location 0x2015, read byte from 0x2016, read byte from 0x2017. Write 32-bits to location 0x4014, the sixth loop of the inner loop.

- Read byte from location 0x2018, read byte from location 0x2019, read byte from 0x201A, read byte from 0x201B. Write 32-bits to location 0x4018, the seventh loop of the inner loop.
- Read byte from location 0x201C, read byte from location 0x201D, read byte from 0x201E, read byte from 0x201F. Write 32-bits to location 0x401C, the last loop of the inner loop.

- DMA engine writes: CFG_SADDRn = 0x2020, CFG_DADDRn = 0x4020, CFG_CLC = 1.
- DMA engine writes: CFG_CSRn[ACTIVE] = 0.
- The channel retires, one loop of the outer loop finish. The DMA goes idle or services the next channel.
- Second hardware request, peripheral assert DMA requests for channel service.
- The channel is selected by arbitration for servicing.
- DMA engine writes: CFG_CSRn[DONE] = 0, CFG_CSRn[START] = 0, CFG_CSRn[ACTIVE] = 1.
- DMA engine reads: channel CFGn data from local memory to the internal register file.
- The source to destination transfers are executed in the following steps:
  - Read byte from location 0x2020, read byte from location 0x2021, read byte from 0x2022, read byte from 0x2023. Write 32-bits to location 0x4020, the first loop of the inner loop.
  - Read byte from location 0x2024, read byte from location 0x2025, read byte from 0x2026, read byte from 0x2027. Write 32-bits to location 0x4024, the second loop of the inner loop.
  - Read byte from location 0x2028, read byte from location 0x2029, read byte from 0x202A, read byte from 0x202B. Write 32-bits to location 0x4028, the third loop of the inner loop.
  - Read byte from location 0x202C, read byte from location 0x202D, read byte from 0x202E, read byte from 0x202F. Write 32-bits to location 0x402C, the fourth loop of the inner loop.
  - Read byte from location 0x2030, read byte from location 0x2031, read byte from 0x2032, read byte from 0x2033. Write 32-bits to location 0x4030, the fifth loop of the inner loop.
  - Read byte from location 0x2034, read byte from location 0x2035, read byte from 0x2036, read byte from 0x2037. Write 32-bits to location 0x4034, the sixth loop of the inner loop.
  - Read byte from location 0x2038, read byte from location 0x2039, read byte from 0x203A, read byte from 0x203B. Write 32-bits to location 0x4038, the seventh loop of the inner loop.
  - Read byte from location 0x203C, read byte from location 0x203D, read byte from 0x203E, read byte from 0x203F. Write 32-bits to location 0x403C, the last loop of the inner loop.
- DMA engine writes: CFG_SADDRn = 0x2000, CFG_DADDRn = 0x4000, CFG_CLCn = 2 (CFG_BLCn).
- DMA engine writes: CFG_CSRn[ACTIVE] = 0, CFG_CSRn[DONE] = 1, INT[n] = 1.
- The channel retires, and the outer loop is complete. The DMA goes idle or services the next channel.

### 17.5.5 Address Modulo Feature

The modulo feature of the DMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. SMOD or DMOD is a 5-bit field for the source address or destination address in the CFG, and it specifies which lower address bits increment from their original value after the address + offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x7654321x) retain their original value. In this example the source address is set to 0x76543210, the offset is set to 4 bytes and the MOD field is set to 4, allowing for a 2^4 bytes (16-byte) size queue.

**Table 48. Modulo example**

| Transfer Number | Address |
|---|---|
| 1 | 0x76543210 |
| 2 | 0x76543214 |
| 3 | 0x76543218 |
| 4 | 0x7654321C |
| 5 | 0x76543210 |
| 6 | 0x76543214 |

### 17.5.6 Address Un-align Modulo Feature

The un-align modulo feature of the DMA provides the ability to transfer data from a solid target address per a service request. The address is decided by DUMOx[SUMO] or DUMOx[DUMO].

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x7654321x) retain their original value. In this example the source address is set to 0x76543210, the offset is set to 4 bytes and the MOD field is set to 4, then the un-align modulo offset registers of source or destination are set to 0x8, the CFG_NBYTESn also needs to be set to 12 bytes, so the address can return to the source start address per reaching 0x76543218.

**Table 49. Un-align modulo example**

| Transfer Number | Address |
|---|---|
| 1 | 0x76543210 |
| 2 | 0x76543214 |
| 3 | 0x76543218 |
| 4 | 0x76543210 |
| 5 | 0x76543214 |
| 6 | 0x76543218 |

### 17.5.7 DMA_CFG ECC Gasket

The DMA_CFG RAM with ECC protection is able to support single-bit correction and double-bit detection, and also supports both address and data ECC.

For the DMA_CFG ECC self-test, using EIM DMA_CFG_ECC injection channel to inject errors. The error types include single-bit data errors, multi-bit data errors, data_chkbit errors, single-bit address errors, and address_chkbit errors. Any ECC error will trigger an ERM interrupt to the system.

### 17.5.8 DMA Lockstep Monitor

The DMA lockstep monitor function is used for function safety during DMA transfer. The DMA lockstep can monitor the DMA engine.

For the DMA lockstep monitor self-test, using EIM DMA lockstep injection channel to set or clear lockstep errors. Any error will trigger an error to the system.

### 17.5.9 DMAMUX

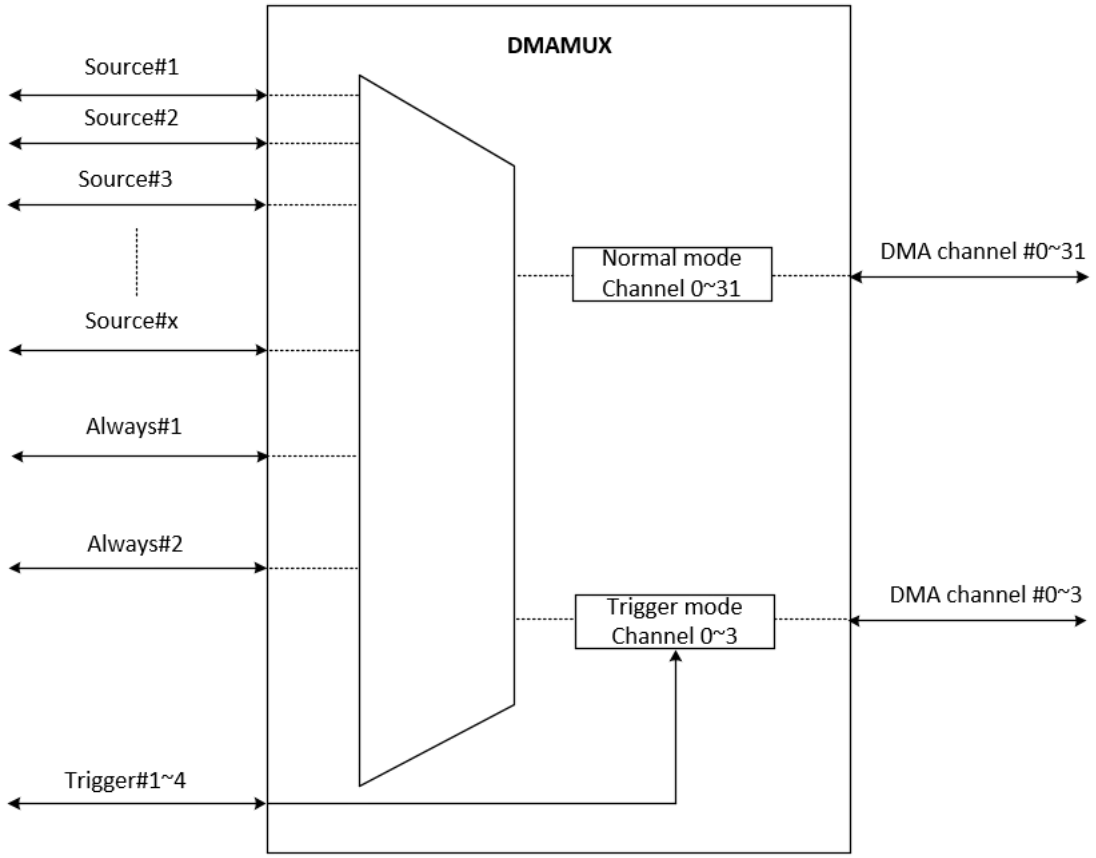
The primary purpose of the DMAMUX is to provide flexibility in the system's use of the available DMA channels. The DMAMUX configuration is a static procedure done during executing the system boot code.

#### 17.5.9.1 DMAMUX Channels Modes

Functionally, the DMAMUX channels support two modes: all channels that implement only the normal routing functionality or the first 4 channels that implement the normal routing functionality plus periodic triggering capability. The first 4 channels provide a periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames, or packets at fixed intervals without the need for processor intervention.

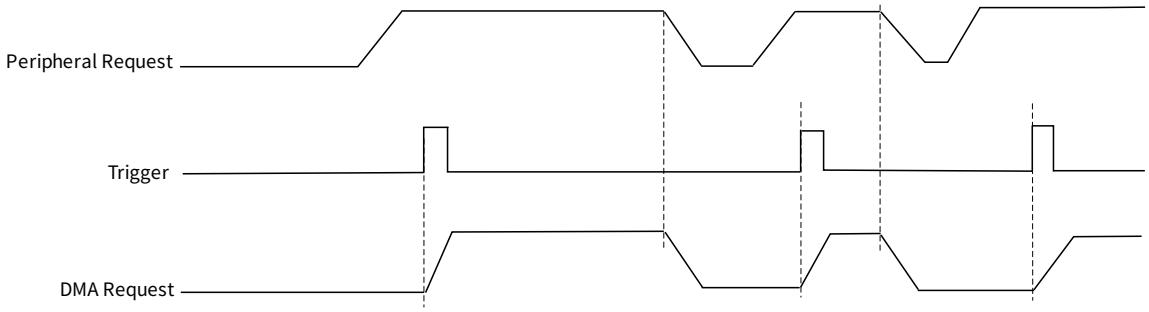
The periodic triggering interval is done via configuring the external periodic timer, such as the external periodic interrupt timer (FCPIT). DMAMUX triggered channels are as follows:

**Figure 32. DMAMUX triggered channels**




On the transmit side of some peripherals, the channel trigger capability usually is used to activate a DMA transfer regularly without intervention from the processor. This trigger works by gating the request from the peripheral to the DMA until a trigger event has been detected. After the DMA request has been serviced, the peripheral will deassert its request, effectively resetting the gating mechanism until the peripheral reasserts its request and the next trigger event is detected. So, if a trigger is detected, but the peripheral is not asserting the request, then that trigger will be ignored. The channel triggering event is as follows:

**Figure 33. DMAMUX channel triggering event**




The triggering mechanism can be used for any peripheral to request DMA transfer, especially in two cases as below:

- Periodically polling external devices on a particular bus:
  For FCIIC, the transmit side of an FCIIC is assigned to a DMA channel with a trigger, as described above. After it has been set up, the FCIIC will request DMA transfers, presumably from memory, as long as its transmit buffer is empty. By using a trigger on this channel, the FCIIC transfers can be automatically performed every 5 µs (as an example). On the receive side of the FCIIC, the FCIIC and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.
- Using the GPIO ports to drive or sample waveforms:
  For GPIO ports receive side, configure the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. For GPIO ports transmit side, by using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

#### *17.5.9.2 DMAMUX Request Source*

DMA request sources mainly include the hardware peripherals and two additional always-enabled sources. Peripherals can control the flow of data during DMA transfers, the always-enabled sources provide no such "throttling" of the data transfers. These always-enabled sources are mainly used in the following cases:

- Performing DMA transfers to/from GPIO — Moving data from/to one or more GPIO pins, either unthrottled (that is, as fast as possible), or periodically (using the DMAMUX triggering capability).
- Performing DMA transfers from memory to memory — Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Performing DMA transfers from memory to the external bus, or vice-versa — Similar to memory-to-memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation — Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, an always-enabled DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the inner loop require that a new start event be sent. This can either be a new software activation, or a transfer request from the DMA channel MUX. The options for doing this are:

- Transfer all data in a single inner loop.
  By configuring the DMA to transfer all the data in a single inner loop (that is, outer loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that

the DMA transfer will impose on the system. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use explicit software reactivation.
  In this option, the DMA is configured to transfer the data using both inner and outer loops, but the processor is required to reactivate the channel by writing to the DMA registers after every inner loop. For this option, the DMA channel must be disabled in the DMA channel MUX.
- Use an always-enabled DMA source.
  In this option, the DMA is configured to transfer the data using both inner and outer loops, and the DMA channel MUX does the channel reactivation. For this option, the DMA channel should be enabled and point to an "always enabled" source. Note that the reactivation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

#### *17.5.9.3 DMAMUX Configure*

The steps to enable a request with periodic triggering capability for the first 4 DMAMUX channels:

1) Select one DMA channel n to service for the request source.
2) Clear the CHCFGn[ENBL] and CHTRG[TRIGn] fields of the DMAMUX register.
3) Configure the DMA properly. The DMA channel may be enabled at this step.
4) Configure the corresponding periodic timer.
5) Program the CHTRG[TRIGn] register.
6) Program the CHCFGn[SOURCE] to select the source to be routed to the DMA channel. Program the CHCFGn register to set CHCFGn[ENBL].

To configure request source #6 transmit for use with DMA channel 2, with periodic triggering capability:

1) Write 0x00 to CHCFG2 register.
2) Configure channel 2 in the DMA, including enabling the channel.
3) Configure a timer for the desired trigger interval.
4) Write 0x4 to CHTRG register.
5) Write 0x86 to CHCFG2 register.

To enable a source, without periodic triggering:

1) Select one DMA channel n to service the request source.
2) Clear the CHCFGn[ENBL] and CHTRG[TRIGn] fields of the DMAMUX register.
3) Configure the DMA properly. The DMA channel may be enabled at this step.
4) Program the CHCFGn[SOURCE] to select the source to be routed to the DMA channel. Program the CHCFGn register to set CHCFGn[ENBL].

To configure source #6 transmit for use with DMA channel 2, with no periodic triggering capability:

1) Write 0x00 to CHCFG2 register.
2) Configure channel 2 in the DMA, including enabling the channel.
3) Write 0x86 to CHCFG2 register.

To disable a request source: a particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Switch the source of a DMA channel as follow steps:

1) Disable the DMA channel in the DMA and reconfigure the channel CHCFGn[SOURCE] for the new source.

2) Clear the CHCFGn[ENBL] bit and CHTRG[TRIGn] bits of the DMA channel.
3) Select the source to be routed to the DMA channel. Write to the corresponding CHCFGn[SOURCE], ensuring that the CHCFGn[ENBL] and CHTRG[TRIGn] fields are set.

To switch DMA channel 8 from source #6 transmit to source #7 transmit:

1) In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2) Write 0x00 to CHCFG8.
3) Write 0x87 to CHCFG8.

## 17.6 DMA Register Definition

DMA base address : 4000_8000h

**Register Overview - DMA**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | DMA_CR (Control Register) | 32 | RW | 0000_0400h | page |
| **4h** | DMA_ES (Error Status Register) | 32 | RO | 0000_0000h | page |
| **Ch** | DMA_ERQ (Enable Request Register) | 32 | RW | 0000_0000h | page |
| **14h** | DMA_EEI (Enable Error Interrupt Register) | 32 | RW | 0000_0000h | page |
| **18h** | DMA_CEEI (Clear Enable Error Interrupt Register) | 8 | WO | 00h | page |
| **19h** | DMA_SEEI (Set Enable Error Interrupt Register) | 8 | WO | 00h | page |
| **1Ah** | DMA_CERQ (Clear Enable Request Register) | 8 | WO | 00h | page |
| **1Bh** | DMA_SERQ (Set Enable Request Register) | 8 | WO | 00h | page |
| **1Ch** | DMA_CDNE (Clear DONE Status Bit Register) | 8 | WO | 00h | page |
| **1Dh** | DMA_SSRT (Set START Bit Register) | 8 | WO | 00h | page |
| **1Eh** | DMA_CERR (Clear Error Register) | 8 | WO | 00h | page |
| **1Fh** | DMA_CINT (Clear Interrupt Request Register) | 8 | WO | 00h | page |
| **24h** | DMA_INT (Interrupt Request Register) | 32 | W1C | 0000_0000h | page |
| **2Ch** | DMA_ERR (Error Register) | 32 | W1C | 0000_0000h | page |
| **34h** | DMA_HRS (Hardware Request Status Register) | 32 | RO | 0000_0000h | page |
| **48h** | DMA_DUME0 (Un-align Modulo Enable Register 0) | 32 | RW | 0000_0000h | page |
| **50h+n*4h** | DMA_DUMOn (Unalign Modulo N Offset Register) | 32 | RW | 0000_0000h | page |
| **100h** | DMA_DCHPRI3 (Channel Priority Register) | 8 | RW | 03h | page |
| **101h** | DMA_DCHPRI2 (Channel Priority Register) | 8 | RW | 02h | page |
| **102h** | DMA_DCHPRI1 (Channel Priority Register) | 8 | RW | 01h | page |

| | | | | | |
|---|---|---|---|---|---|
| **103h** | DMA_DCHPRI0 (Channel Priority Register) | 8 | RW | 00h | page |
| **104h** | DMA_DCHPRI7 (Channel Priority Register) | 8 | RW | 07h | page |
| **105h** | DMA_DCHPRI6 (Channel Priority Register) | 8 | RW | 06h | page |
| **106h** | DMA_DCHPRI5 (Channel Priority Register) | 8 | RW | 05h | page |
| **107h** | DMA_DCHPRI4 (Channel Priority Register) | 8 | RW | 04h | page |
| **108h** | DMA_DCHPRI11 (Channel Priority Register) | 8 | RW | 0Bh | page |
| **109h** | DMA_DCHPRI10 (Channel Priority Register) | 8 | RW | 0Ah | page |
| **10Ah** | DMA_DCHPRI9 (Channel Priority Register) | 8 | RW | 09h | page |
| **10Bh** | DMA_DCHPRI8 (Channel Priority Register) | 8 | RW | 08h | page |
| **10Ch** | DMA_DCHPRI15 (Channel Priority Register) | 8 | RW | 0Fh | page |
| **10Dh** | DMA_DCHPRI14 (Channel Priority Register) | 8 | RW | 0Eh | page |
| **10Eh** | DMA_DCHPRI13 (Channel Priority Register) | 8 | RW | 0Dh | page |
| **10Fh** | DMA_DCHPRI12 (Channel Priority Register) | 8 | RW | 0Ch | page |
| **1000h+n*20h** | DMA_CFG_SADDRn (CFG Source Address Register) | 32 | RW | Undefined | page |
| **1004h+n*20h** | DMA_CFG_SOFFn (CFG Signed Source Address Offset Register) | 16 | RW | Undefined | page |
| **1006h+n*20h** | DMA_CFG_ATTRn (CFG Transfer Attributes Register) | 16 | RW | Undefined | page |
| **1008h+n*20h** | DMA_CFG_NBYTES_ILNOn (CFG Inner Byte Count Inner Loop Mapping Disabled Register) | 32 | RW | Undefined | page |
| **1008h+n*20h** | DMA_CFG_NBYTES_ILOFFNOn (CFG Signed Inner Loop Offset Inner Loop Mapping Enabled and Offset Disabled Register) | 32 | RW | Undefined | page |
| **1008h+n*20h** | DMA_CFG_NBYTES_ILOFFYESn (CFG Signed Inner Loop Offset Inner Loop Mapping and Offset Enabled Register) | 32 | RW | Undefined | page |
| **100Ch+n*20h** | DMA_CFG_SLASTn (CFG Last Source Address Adjustment Register) | 32 | RW | Undefined | page |
| **1010h+n*20h** | DMA_CFG_DADDRn (CFG Destination Address Register) | 32 | RW | Undefined | page |
| **1014h+n*20h** | DMA_CFG_DOFFn (CFG Signed Destination Address Offset Register) | 16 | RW | Undefined | page |
| **1016h+n*20h** | DMA_CFG_CLC_CHTRGENNOn (CFG Current Inner Loop Trigger, Outer Loop Count Channel Trigger Disabled Register) | 16 | RW | Undefined | page |
| **1016h+n*20h** | DMA_CFG_CLC_CHTRGENYESn (CFG Current Inner Loop Trigger, Outer Loop Count Channel Trigger Enabled Register) | 16 | RW | Undefined | page |
| **1018h+n*20h** | DMA_CFG_DLASTn (CFG Last Destination Address Adjustment Register) | 32 | RW | Undefined | page |

| | | | | | |
|---|---|---|---|---|---|
| **101Ch+n*20h** | DMA_CFG_CSRn (CFG Control and Status Register) | 16 | RW | Undefined | page |
| **101Eh+n*20h** | DMA_CFG_BLC_CHTRGENNOn (CFG Beginning Inner Loop Trigger, Outer Loop Count Channel Trigger Disabled Register) | 16 | RW | Undefined | page |
| **101Eh+n*20h** | DMA_CFG_BLC_CHTRGENYESn (CFG Beginning Inner Loop Trigger, Outer Loop Count Channel Trigger Enabled Register) | 16 | RW | Undefined | page |

### 17.6.1 DMA_CR (Control Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ACTIVE | 0 | | | | | | | | | | | | | CX | ECX |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | GPR0PRI | EILM | CTM | HALT | HOE | 0 | ERCA | DBGS | 0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_CR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**ACTIVE** | RW | DMA Active Status<br>0b - DMA is idle.<br>1b - DMA is executing a channel. |
| **17**<br>**CX** | RW | Cancel Transfer<br>0b - Normal operation<br>1b - Cancel the remaining data transfer. Stop the transferring channel and the inner loop is forced to finish. The cancel is effective after the last write of the current read/write sequence. The CX bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the inner loop was completed. |

| | | |
|---|---|---|
| **16**<br>**ECX** | RW | Error Cancel Transfer<br>0b - Normal operation<br>1b - Cancel the remaining data transfer in the same fashion as the CX bit. Stop the transferring channel and the inner loop is forced to finish. The cancel is effective after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition, thus updating the Error Status register (DMAx_ES) and generating an optional error interrupt. |
| **8**<br>**GPR0PRI** | RW | Channel Group 0 Priority<br>Group 0 priority level when fixed priority group arbitration is enabled. |
| **7**<br>**EILM** | RW | Enable Inner Loop Mapping<br>0b - Disabled the inner loop mapping. DMA_CFG_NBYTES_MLNOn is defined as a 32-bit NBYTES field.<br>1b - Enabled the inner loop mapping. DMA_CFG_NBYTES_MLOFFYESn is redefined as the fields including individual enable, an offset, and the NBYTES field. The individual enable fields allow the inner loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled. |
| **6**<br>**CTM** | RW | Continuous Trigger Mode<br>For only one inner loop per service request, not need use the continuous trigger mode with a channel trigger to itself.<br>0b - Disable continuous trigger mode. Channel arbitration is used for an inner loop channel trigger before being activated again.<br>1b - Enable continuous trigger mode.Channel arbitration is not used for an inner loop channel trigger before being activated again. Upon inner loop completion, the channel activates again if that channel has an inner loop channel trigger enabled and the trigger channel is itself. This effectively applies the inner loop offsets and restarts the next inner loop. |
| **5**<br>**HALT** | RW | Halt DMA Operations<br>0b - Normal operation<br>1b - Stall any new channels start. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared. |
| **4**<br>**HOE** | RW | Halt On Error<br>0b - Normal operation<br>1b - Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared. |
| **2**<br>**ERCA** | RW | Enable Round Robin Channel Arbitration<br>0b - Fixed priority arbitration is used for channel selection.<br>1b - Round robin arbitration is used for channel selection . |

| | | |
|---|---|---|
| **1**<br>**DBGS** | RW | DMA in debug mode stop<br>0b - The DMA continues to operate when in debug mode.<br>1b - The DMA stalls the start of a new channel when in debug mode. Executing channels are allowed to complete. Channel execution resumes when the system exits debug mode or the DBGS bit is cleared. |

### 17.6.2 DMA_ES (Error Status Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VLD | 0 | | | | | | | | | | | | | | ECX |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | GPE | CPE | 0 | ERRCHN | | | | | SAE | SOE | DAE | DOE | NCE | 0 | SBE | DBE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_ES field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**VLD** | RO | VLD<br>Logical OR of all ERR status bits.<br>0b - No ERR bits are set.<br>1b - At least one ERR bit is set indicating a valid error exists that has not been cleared. |
| **16**<br>**ECX** | RO | Transfer Canceled Error<br>0b - No canceled transfers<br>1b - The last recorded entry was a canceled transfer by the error cancel transfer input |
| **15**<br>**GPE** | RO | Group Priority Error<br>0 - No group priority error.<br>1 - The most-recently recorded error was a configuration error among the group priorities. All group priorities are not unique. |
| **14**<br>**CPE** | RO | Channel Priority Error<br>0b - No channel priority error<br>1b - The last recorded error was a channel priorities configuration error. Channel priorities are not unique. |

| | | |
|---|---|---|
| **12-8**<br>**ERRCHN** | RO | Error Channel Number or Canceled Channel Number<br>The channel number of the last recorded error, excluding CPE errors, or last recorded error canceled transfer. |
| **7**<br>**SAE** | RO | Source Address Error<br>0b - No source address configuration error.<br>1b - The last recorded error was a CFG_SADDRn field configuration error. CFG_SADDRn is inconsistent with CFG_ATTRn[SSIZE]. |
| **6**<br>**SOE** | RO | Source Offset Error<br>0b - No source offset configuration error<br>1b - The last recorded error was a CFG_SOFFn field configuration error. CFG_SOFFn is inconsistent with CFG_ATTRn[SSIZE]. |
| **5**<br>**DAE** | RO | Destination Address Error<br>0b - No destination address configuration error<br>1b - The last recorded error was a CFG_DADDRn field configuration error. CFG_DADDRn is inconsistent with CFG_ATTRn[DSIZE]. |
| **4**<br>**DOE** | RO | Destination Offset Error<br>0b - No destination offset configuration error<br>1b - The last recorded error was a CFG_DOFFn field configuration error. CFG_DOFFn is inconsistent with CFG_ATTRn[DSIZE]. |
| **3**<br>**NCE** | RO | NBYTES/CLC Configuration Error<br>0b - No NBYTES/CLC configuration error<br>1b - The last recorded error was CFG_NBYTESn or CFG_CLCn fields configuration error. CFG_NBYTESn is not a multiple of CFG_ATTRn[SSIZE] and CFG_ATTRn[DSIZE], or CFG_CLCn[CLC] is equal to zero, or CFGn_CLC[CHTRGEN] is not equal to CFG_BLCn[CHTRGEN] |
| **1**<br>**SBE** | RO | Source Bus Error<br>0b - No source bus error<br>1b - The last recorded error was a bus error on a source read |
| **0**<br>**DBE** | RO | Destination Bus Error<br>0b - No destination bus error<br>1b - The last recorded error was a bus error on a destination write |

### 17.6.3 DMA_ERQ (Enable Request Register)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ERQ15 | ERQ14 | ERQ13 | ERQ12 | ERQ11 | ERQ10 | ERQ9 | ERQ8 | ERQ7 | ERQ6 | ERQ5 | ERQ4 | ERQ3 | ERQ2 | ERQ1 | ERQ0 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_ERQ field descriptions**

| Field | Type | Description |
|---|---|---|
| **15**<br>**ERQ15** | RW | Enable DMA Request 15<br>0b – Disable the DMA request signal for the channel 15<br>1b – Enable the DMA request signal for the channel 15 |
| **14**<br>**ERQ14** | RW | Enable DMA Request 14<br>0b – Disable the DMA request signal for the channel 14<br>1b – Enable the DMA request signal for the channel 14 |
| **13**<br>**ERQ13** | RW | Enable DMA Request 13<br>0b – Disable the DMA request signal for the channel 13<br>1b – Enable the DMA request signal for the channel 13 |
| **12**<br>**ERQ12** | RW | Enable DMA Request 12<br>0b – Disable the DMA request signal for the channel 12<br>1b – Enable the DMA request signal for the channel 12 |
| **11**<br>**ERQ11** | RW | Enable DMA Request 11<br>0b – Disable the DMA request signal for the channel 11<br>1b – Enable the DMA request signal for the channel 11 |
| **10**<br>**ERQ10** | RW | Enable DMA Request 10<br>0b – Disable the DMA request signal for the channel 10<br>1b – Enable the DMA request signal for the channel 10 |
| **9**<br>**ERQ9** | RW | Enable DMA Request 9<br>0b – Disable the DMA request signal for the channel 9<br>1b – Enable the DMA request signal for the channel 9 |
| **8**<br>**ERQ8** | RW | Enable DMA Request 8<br>0b – Disable the DMA request signal for the channel 8<br>1b – Enable the DMA request signal for the channel 8 |
| **7**<br>**ERQ7** | RW | Enable DMA Request 7<br>0b – Disable the DMA request signal for the channel 7<br>1b – Enable the DMA request signal for the channel 7 |

| | | |
|---|---|---|
| **6**<br>**ERQ6** | RW | Enable DMA Request 6<br>0b – Disable the DMA request signal for the channel 6<br>1b – Enable the DMA request signal for the channel 6 |
| **5**<br>**ERQ5** | RW | Enable DMA Request 5<br>0b – Disable the DMA request signal for the channel 5<br>1b – Enable the DMA request signal for the channel 5 |
| **4**<br>**ERQ4** | RW | Enable DMA Request 4<br>0b – Disable the DMA request signal for the channel 4<br>1b – Enable the DMA request signal for the channel 4 |
| **3**<br>**ERQ3** | RW | Enable DMA Request 3<br>0b – Disable the DMA request signal for the channel 3<br>1b – Enable the DMA request signal for the channel 3 |
| **2**<br>**ERQ2** | RW | Enable DMA Request 2<br>0b – Disable the DMA request signal for the channel 2<br>1b – Enable the DMA request signal for the channel 2 |
| **1**<br>**ERQ1** | RW | Enable DMA Request 1<br>0b – Disable the DMA request signal for the channel 1<br>1b – Enable the DMA request signal for the channel 1 |
| **0**<br>**ERQ0** | RW | Enable DMA Request 0<br>0b – Disable the DMA request signal for the channel 0<br>1b – Enable the DMA request signal for the channel 0 |

### 17.6.4 DMA_EEI (Enable Error Interrupt Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EEI15 | EEI14 | EEI13 | EEI12 | EEI11 | EEI10 | EEI9 | EEI8 | EEI7 | EEI6 | EEI5 | EEI4 | EEI3 | EEI2 | EEI1 | EEI0 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_EEI field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **15**<br>**EEI15** | RW | Enable Error Interrupt 15<br>0b - Disable the error signal for channel 15 to generate an error interrupt<br>1b - Enable the error signal for channel 15 to generate an error interrupt request |
| **14**<br>**EEI14** | RW | Enable Error Interrupt 14<br>0b - Disable the error signal for channel 14 to generate an error interrupt<br>1b - Enable the error signal for channel 14 to generate an error interrupt request |
| **13**<br>**EEI13** | RW | Enable Error Interrupt 13<br>0b - Disable the error signal for channel 13 to generate an error interrupt<br>1b - Enable the error signal for channel 13 to generate an error interrupt request |
| **12**<br>**EEI12** | RW | Enable Error Interrupt 12<br>0b - Disable the error signal for channel 12 to generate an error interrupt<br>1b - Enable the error signal for channel 12 to generate an error interrupt request |
| **11**<br>**EEI11** | RW | Enable Error Interrupt 11<br>0b - Disable the error signal for channel 11 to generate an error interrupt<br>1b - Enable the error signal for channel 11 to generate an error interrupt request |
| **10**<br>**EEI10** | RW | Enable Error Interrupt 10<br>0b - Disable the error signal for channel 10 to generate an error interrupt<br>1b - Enable the error signal for channel 10 to generate an error interrupt request |
| **9**<br>**EEI9** | RW | Enable Error Interrupt 9<br>0b - Disable the error signal for channel 9 to generate an error interrupt<br>1b - Enable the error signal for channel 9 to generate an error interrupt request |
| **8**<br>**EEI8** | RW | Enable Error Interrupt 8<br>0b - Disable the error signal for channel 8 to generate an error interrupt<br>1b - Enable the error signal for channel 8 to generate an error interrupt request |
| **7**<br>**EEI7** | RW | Enable Error Interrupt 7<br>0b - Disable the error signal for channel 7 to generate an error interrupt<br>1b - Enable the error signal for channel 7 to generate an error interrupt request |

| | | |
|---|---|---|
| **6**<br>**EEI6** | RW | Enable Error Interrupt 6<br>0b - Disable the error signal for channel 6 to generate an error interrupt<br>1b - Enable the error signal for channel 6 to generate an error interrupt request |
| **5**<br>**EEI5** | RW | Enable Error Interrupt 5<br>0b - Disable the error signal for channel 5 to generate an error interrupt<br>1b - Enable the error signal for channel 5 to generate an error interrupt request |
| **4**<br>**EEI4** | RW | Enable Error Interrupt 4<br>0b - Disable the error signal for channel 4 to generate an error interrupt<br>1b - Enable the error signal for channel 4 to generate an error interrupt request |
| **3**<br>**EEI3** | RW | Enable Error Interrupt 3<br>0b - Disable the error signal for channel 3 to generate an error interrupt<br>1b - Enable the error signal for channel 3 to generate an error interrupt request |
| **2**<br>**EEI2** | RW | Enable Error Interrupt 2<br>0b - Disable the error signal for channel 2 to generate an error interrupt<br>1b - Enable the error signal for channel 2 to generate an error interrupt request |
| **1**<br>**EEI1** | RW | Enable Error Interrupt 1<br>0b - Disable the error signal for channel 1 to generate an error interrupt<br>1b - Enable the error signal for channel 1 to generate an error interrupt request |
| **0**<br>**EEI0** | RW | Enable Error Interrupt 0<br>0b - Disable the error signal for channel 0 to generate an error interrupt<br>1b - Enable the error signal for channel 0 to generate an error interrupt request |

### 17.6.5 DMA_CEEI (Clear Enable Error Interrupt Register)

Offset : 18h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | | | |
| W | NOP | CAEE | | CEEI | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_CEEI field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **7**<br>**NOP** | WO | No Operation Enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| **6**<br>**CAEE** | WO | Clear All Enable Error Interrupts<br>0b - Clear only the EEI bit specified in the CEEI field<br>1b - Clear all bits in EEI |
| **4-0**<br>**CEEI** | WO | Clear Enable Error Interrupt<br>Clears the corresponding bit in EEI |

### 17.6.6 DMA_SEEI (Set Enable Error Interrupt Register)

Offset : 19h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | | | |
| W | NOP | SAEE | | SEEI | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_SEEI field descriptions**

| Field | Type | Description |
|---|---|---|
| **7**<br>**NOP** | WO | No Operation Enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| **6**<br>**SAEE** | WO | Sets All Enable Error Interrupts<br>0b - Set only the EEI bit specified in the SEEI field.<br>1b - Sets all bits in EEI |
| **4-0**<br>**SEEI** | WO | Set Enable Error Interrupt<br>Sets the corresponding bit in EEI |

### 17.6.7 DMA_CERQ (Clear Enable Request Register)

Offset : 1Ah

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | | | |
| W | NOP | CAER | | CERQ | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_CERQ field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **7**<br>**NOP** | WO | No Operation Enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| **6**<br>**CAER** | WO | Clear All Enable Requests<br>0b - Clear only the ERQ bit specified in the CERQ field<br>1b - Clear all bits in ERQ |
| **4-0**<br>**CERQ** | WO | Clear Enable Request<br>Clears the corresponding bit in ERQ. |

### 17.6.8 DMA_SERQ (Set Enable Request Register)

Offset : 1Bh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | | | |
| W | NOP | SAER | | SERQ | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_SERQ field descriptions**

| Field | Type | Description |
|---|---|---|
| **7**<br>**NOP** | WO | No Operation Enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| **6**<br>**SAER** | WO | Set All Enable Requests<br>0b - Set only the ERQ bit specified in the SERQ field<br>1b - Set all bits in ERQ |
| **4-0**<br>**SERQ** | WO | Set Enable Request<br>Sets the corresponding bit in ERQ. |

### 17.6.9 DMA_CDNE (Clear DONE Status Bit Register)

Offset : 1Ch

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | | | |
| W | NOP | CADN | | CDNE | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_CDNE field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **7**<br>**NOP** | WO | No Operation Enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| **6**<br>**CADN** | WO | Clears All DONE Bits<br>0b - Clears only the CFG_CSRn[DONE] bit specified in the CDNE field<br>1b - Clears all bits in CFG_CSRn[DONE] |
| **4-0**<br>**CDNE** | WO | Clear DONE Bit<br>Clears the corresponding bit in CFG_CSRn[DONE] |

### 17.6.10 DMA_SSRT (Set START Bit Register)

Offset : 1Dh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | | | |
| W | NOP | SAST | | SSRT | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_SSRT field descriptions**

| Field | Type | Description |
|---|---|---|
| **7**<br>**NOP** | WO | No Operation Enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| **6**<br>**SAST** | WO | Set All START Bits (activates all channels)<br>0b - Set only the CFG_CSRn[START] bit specified in the SSRT field<br>1b - Set all bits in CFG_CSRn[START] |
| **4-0**<br>**SSRT** | WO | Set START Bit<br>Sets the corresponding bit in CFG_CSRn[START] |

### 17.6.11 DMA_CERR (Clear Error Register)

Offset : 1Eh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | | | |
| W | NOP | CAEI | | CERR | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_CERR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **7**<br>**NOP** | WO | No Operation Enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| **6**<br>**CAEI** | WO | Clear All Error Indicators<br>0b - Clear only the ERR bit specified in the CERR field<br>1b - Clear all bits in ERR |
| **4-0**<br>**CERR** | WO | Clear Error Indicator<br>Clears the corresponding bit in ERR |

### 17.6.12 DMA_CINT (Clear Interrupt Request Register)

Offset : 1Fh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | | | |
| W | NOP | CAIR | | CINT | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_CINT field descriptions**

| Field | Type | Description |
|---|---|---|
| **7**<br>**NOP** | WO | No Operation Enable<br>0b - Normal operation<br>1b - No operation, ignore the other bits in this register |
| **6**<br>**CAIR** | WO | Clear All Interrupt Requests<br>0b - Clear only the INT bit specified in the CINT field<br>1b - Clear all bits in INT |
| **4-0**<br>**CINT** | WO | Clear Interrupt Request<br>Clears the corresponding bit in INT |

### 17.6.13 DMA_INT (Interrupt Request Register)

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INT15 | INT14 | INT13 | INT12 | INT11 | INT10 | INT9 | INT8 | INT7 | INT6 | INT5 | INT4 | INT3 | INT2 | INT1 | INT0 |
| W | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_INT field descriptions**

| Field | Type | Description |
|---|---|---|
| **15**<br>**INT15** | W1C | Interrupt Request 15<br>0b - The interrupt request for channel 15 is cleared<br>1b - The interrupt request for channel 15 is active |
| **14**<br>**INT14** | W1C | Interrupt Request 14<br>0b - The interrupt request for channel 14 is cleared<br>1b - The interrupt request for channel 14 is active |
| **13**<br>**INT13** | W1C | Interrupt Request 13<br>0b - The interrupt request for channel 13 is cleared<br>1b - The interrupt request for channel 13 is active |
| **12**<br>**INT12** | W1C | Interrupt Request 12<br>0b - The interrupt request for channel 12 is cleared<br>1b - The interrupt request for channel 12 is active |
| **11**<br>**INT11** | W1C | Interrupt Request 11<br>0b - The interrupt request for channel 11 is cleared<br>1b - The interrupt request for channel 11 is active |
| **10**<br>**INT10** | W1C | Interrupt Request 10<br>0b - The interrupt request for channel 10 is cleared<br>1b - The interrupt request for channel 10 is active |
| **9**<br>**INT9** | W1C | Interrupt Request 9<br>0b - The interrupt request for channel 9 is cleared<br>1b - The interrupt request for channel 9 is active |
| **8**<br>**INT8** | W1C | Interrupt Request 8<br>0b - The interrupt request for channel 8 is cleared<br>1b - The interrupt request for channel 8 is active |
| **7**<br>**INT7** | W1C | Interrupt Request 7<br>0b - The interrupt request for channel 7 is cleared<br>1b - The interrupt request for channel 7 is active |
| **6**<br>**INT6** | W1C | Interrupt Request 6<br>0b - The interrupt request for channel 6 is cleared<br>1b - The interrupt request for channel 6 is active |

| | | |
|---|---|---|
| **5**<br>**INT5** | W1C | Interrupt Request 5<br>0b - The interrupt request for channel 5 is cleared<br>1b - The interrupt request for channel 5 is active |
| **4**<br>**INT4** | W1C | Interrupt Request 4<br>0b - The interrupt request for channel 4 is cleared<br>1b - The interrupt request for channel 4 is active |
| **3**<br>**INT3** | W1C | Interrupt Request 3<br>0b - The interrupt request for channel 3 is cleared<br>1b - The interrupt request for channel 3 is active |
| **2**<br>**INT2** | W1C | Interrupt Request 2<br>0b - The interrupt request for channel 2 is cleared<br>1b - The interrupt request for channel 2 is active |
| **1**<br>**INT1** | W1C | Interrupt Request 1<br>0b - The interrupt request for channel 1 is cleared<br>1b - The interrupt request for channel 1 is active |
| **0**<br>**INT0** | W1C | Interrupt Request 0<br>0b - The interrupt request for channel 0 is cleared<br>1b - The interrupt request for channel 0 is active |

### 17.6.14 DMA_ERR (Error Register)

Offset : 2Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ERR15 | ERR14 | ERR13 | ERR12 | ERR11 | ERR10 | ERR9 | ERR8 | ERR7 | ERR6 | ERR5 | ERR4 | ERR3 | ERR2 | ERR1 | ERR0 |
| W | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_ERR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **15**<br>**ERR15** | W1C | Error In Channel 15<br>0b – No error occurred in channel 15<br>1b – An error occurred in channel 15 |
| **14**<br>**ERR14** | W1C | Error In Channel 14<br>0b – No error occurred in channel 14<br>1b – An error occurred in channel 14 |
| **13**<br>**ERR13** | W1C | Error In Channel 13<br>0b – No error occurred in channel 13<br>1b – An error occurred in channel 13 |
| **12**<br>**ERR12** | W1C | Error In Channel 12<br>0b – No error occurred in channel 12<br>1b – An error occurred in channel 12 |
| **11**<br>**ERR11** | W1C | Error In Channel 11<br>0b – No error occurred in channel 11<br>1b – An error occurred in channel 11 |
| **10**<br>**ERR10** | W1C | Error In Channel 10<br>0b – No error occurred in channel 10<br>1b – An error occurred in channel 10 |
| **9**<br>**ERR9** | W1C | Error In Channel 9<br>0b – No error occurred in channel 9<br>1b – An error occurred in channel 9 |
| **8**<br>**ERR8** | W1C | Error In Channel 8<br>0b – No error occurred in channel 8<br>1b – An error occurred in channel 8 |
| **7**<br>**ERR7** | W1C | Error In Channel 7<br>0b – No error occurred in channel 7<br>1b – An error occurred in channel 7 |
| **6**<br>**ERR6** | W1C | Error In Channel 6<br>0b – No error occurred in channel 6<br>1b – An error occurred in channel 6 |
| **5**<br>**ERR5** | W1C | Error In Channel 5<br>0b – No error occurred in channel 5<br>1b – An error occurred in channel 5 |
| **4**<br>**ERR4** | W1C | Error In Channel 4<br>0b – No error occurred in channel 4<br>1b – An error occurred in channel 4 |

| | | |
|---|---|---|
| **3**<br>**ERR3** | W1C | Error In Channel 3<br>0b – No error occurred in channel 3<br>1b – An error occurred in channel 3 |
| **2**<br>**ERR2** | W1C | Error In Channel 2<br>0b – No error occurred in channel 2<br>1b – An error occurred in channel 2 |
| **1**<br>**ERR1** | W1C | Error In Channel 1<br>0b – No error occurred in channel 1<br>1b – An error occurred in channel 1 |
| **0**<br>**ERR0** | W1C | Error In Channel 0<br>0b – No error occurred in channel 0<br>1b – An error occurred in channel 0 |

### 17.6.15 DMA_HRS (Hardware Request Status Register)

Offset : 34h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | HRS15 | HRS14 | HRS13 | HRS12 | HRS11 | HRS10 | HRS9 | HRS8 | HRS7 | HRS6 | HRS5 | HRS4 | HRS3 | HRS2 | HRS1 | HRS0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_HRS field descriptions**

| Field | Type | Description |
|---|---|---|
| **15**<br>**HRS15** | RO | Hardware Request Status Channel 15<br>The HRS bits in channel 15 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 15<br>1b – Assert hardware service request for channel 15 |

| | | |
|---|---|---|
| **14**<br>**HRS14** | RO | Hardware Request Status Channel 14<br>The HRS bits in channel 14 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 14<br>1b – Assert hardware service request for channel 14 |
| **13**<br>**HRS13** | RO | Hardware Request Status Channel 13<br>The HRS bits in channel 13 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 13<br>1b – Assert hardware service request for channel 13 |
| **12**<br>**HRS12** | RO | Hardware Request Status Channel 12<br>The HRS bits in channel 12 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 12<br>1b – Assert hardware service request for channel 12 |
| **11**<br>**HRS11** | RO | Hardware Request Status Channel 11<br>The HRS bits in channel 11 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 11<br>1b – Assert hardware service request for channel 11 |
| **10**<br>**HRS10** | RO | Hardware Request Status Channel 10<br>The HRS bits in channel 10 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 10<br>1b – Assert hardware service request for channel 10 |
| **9**<br>**HRS9** | RO | Hardware Request Status Channel 9<br>The HRS bits in channel 9 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 9<br>1b – Assert hardware service request for channel 9 |

| | | |
|---|---|---|
| **8**<br>**HRS8** | RO | Hardware Request Status Channel 8<br>The HRS bits in channel 8 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 8<br>1b – Assert hardware service request for channel 8 |
| **7**<br>**HRS7** | RO | Hardware Request Status Channel 7<br>The HRS bits in channel 7 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 7<br>1b – Assert hardware service request for channel 7 |
| **6**<br>**HRS6** | RO | Hardware Request Status Channel 6<br>The HRS bits in channel 6 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 6<br>1b – Assert hardware service request for channel 6 |
| **5**<br>**HRS5** | RO | Hardware Request Status Channel 5<br>The HRS bits in channel 5 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 5<br>1b – Assert hardware service request for channel 5 |
| **4**<br>**HRS4** | RO | Hardware Request Status Channel 4<br>The HRS bits in channel 4 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 4<br>1b – Assert hardware service request for channel 4 |
| **3**<br>**HRS3** | RO | Hardware Request Status Channel 3<br>The HRS bits in channel 3 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 3<br>1b – Assert hardware service request for channel 3 |

| | | |
|---|---|---|
| **2**<br>**HRS2** | RO | Hardware Request Status Channel 2<br>The HRS bits in channel 2 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 2<br>1b – Assert hardware service request for channel 2 |
| **1**<br>**HRS1** | RO | Hardware Request Status Channel 1<br>The HRS bits in channel 1 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 1<br>1b – Assert hardware service request for channel 1 |
| **0**<br>**HRS0** | RO | Hardware Request Status Channel 0<br>The HRS bits in channel 0 keep assertion when existing a hardware service request. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.<br>0b – No hardware service request for channel 0<br>1b – Assert hardware service request for channel 0 |

### 17.6.16 DMA_DUME0 (Un-align Modulo Enable Register 0)

Offset : 48h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SUME 15 | DUME 15 | SUME 14 | DUME 14 | SUME 13 | DUME 13 | SUME 12 | DUME 12 | SUME 11 | DUME 11 | SUME 10 | DUME 10 | SUME 9 | DUME 9 | SUME 8 | DUME 8 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SUME 7 | DUME 7 | SUME 6 | DUME 6 | SUME 5 | DUME 5 | SUME 4 | DUME 4 | SUME 3 | DUME 3 | SUME 2 | DUME 2 | SUME 1 | DUME 1 | SUME 0 | DUME 0 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_DUME0 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31**<br>**SUME15** | RW | Channel 15 source address un-align modulo enable<br>0b - Disable channel 15 source address un-align modulo<br>1b - Enable channel 15 source address un-align modulo |
| **30**<br>**DUME15** | RW | Channel 15 destination address un-align modulo enable<br>0b - Disable channel 15 destination address un-align modulo<br>1b - Enable channel 15 destination address un-align modulo |
| **29**<br>**SUME14** | RW | Channel 14 source address un-align modulo enable<br>0b - Disable channel 14 source address un-align modulo<br>1b - Enable channel 14 source address un-align modulo |
| **28**<br>**DUME14** | RW | Channel 14 destination address un-align modulo enable<br>0b - Disable channel 14 destination address un-align modulo<br>1b - Enable channel 14 destination address un-align modulo |
| **27**<br>**SUME13** | RW | Channel 13 source address un-align modulo enable<br>0b - Disable channel 13 source address un-align modulo<br>1b - Enable channel 13 source address un-align modulo |
| **26**<br>**DUME13** | RW | Channel 13 destination address un-align modulo enable<br>0b - Disable channel 13 destination address un-align modulo<br>1b - Enable channel 13 destination address un-align modulo |
| **25**<br>**SUME12** | RW | Channel 12 source address un-align modulo enable<br>0b - Disable channel 12 source address un-align modulo<br>1b - Enable channel 12 source address un-align modulo |
| **24**<br>**DUME12** | RW | Channel 12 destination address un-align modulo enable<br>0b - Disable channel 12 destination address un-align modulo<br>1b - Enable channel 12 destination address un-align modulo |
| **23**<br>**SUME11** | RW | Channel 11 source address un-align modulo enable<br>0b - Disable channel 11 source address un-align modulo<br>1b - Enable channel 11 source address un-align modulo |
| **22**<br>**DUME11** | RW | Channel 11 destination address un-align modulo enable<br>0b - Disable channel 11 destination address un-align modulo<br>1b - Enable channel 11 destination address un-align modulo |
| **21**<br>**SUME10** | RW | Channel 10 source address un-align modulo enable<br>0b - Disable channel 10 source address un-align modulo<br>1b - Enable channel 10 source address un-align modulo |
| **20**<br>**DUME10** | RW | Channel 10 destination address un-align modulo enable<br>0b - Disable channel 10 destination address un-align modulo<br>1b - Enable channel 10 destination address un-align modulo |

| | | |
|---|---|---|
| **19**<br>**SUME9** | RW | Channel 9 source address un-align modulo enable<br>0b - Disable channel 9 source address un-align modulo<br>1b - Enable channel 9 source address un-align modulo |
| **18**<br>**DUME9** | RW | Channel 9 destination address un-align modulo enable<br>0b - Disable channel 9 destination address un-align modulo<br>1b - Enable channel 9 destination address un-align modulo |
| **17**<br>**SUME8** | RW | Channel 8 source address un-align modulo enable<br>0b - Disable channel 8 source address un-align modulo<br>1b - Enable channel 8 source address un-align modulo |
| **16**<br>**DUME8** | RW | Channel 8 destination address un-align modulo enable<br>0b - Disable channel 8 destination address un-align modulo<br>1b - Enable channel 8 destination address un-align modulo |
| **15**<br>**SUME7** | RW | Channel 7 source address un-align modulo enable<br>0b - Disable channel 7 source address un-align modulo<br>1b - Enable channel 7 source address un-align modulo |
| **14**<br>**DUME7** | RW | Channel 7 destination address un-align modulo enable<br>0b - Disable channel 7 destination address un-align modulo<br>1b - Enable channel 7 destination address un-align modulo |
| **13**<br>**SUME6** | RW | Channel 6 source address un-align modulo enable<br>0b - Disable channel 6 source address un-align modulo<br>1b - Enable channel 6 source address un-align modulo |
| **12**<br>**DUME6** | RW | Channel 6 destination address un-align modulo enable<br>0b - Disable channel 6 destination address un-align modulo<br>1b - Enable channel 6 destination address un-align modulo |
| **11**<br>**SUME5** | RW | Channel 5 source address un-align modulo enable<br>0b - Disable channel 5 source address un-align modulo<br>1b - Enable channel 5 source address un-align modulo |
| **10**<br>**DUME5** | RW | Channel 5 destination address un-align modulo enable<br>0b - Disable channel 5 destination address un-align modulo<br>1b - Enable channel 5 destination address un-align modulo |
| **9**<br>**SUME4** | RW | Channel 4 source address un-align modulo enable<br>0b - Disable channel 4 source address un-align modulo<br>1b - Enable channel 4 source address un-align modulo |
| **8**<br>**DUME4** | RW | Channel 4 destination address un-align modulo enable<br>0b - Disable channel 4 destination address un-align modulo<br>1b - Enable channel 4 destination address un-align modulo |

| | | |
|---|---|---|
| **7**<br>**SUME3** | RW | Channel 3 source address un-align modulo enable<br>0b - Disable channel 3 source address un-align modulo<br>1b - Enable channel 3 source address un-align modulo |
| **6**<br>**DUME3** | RW | Channel 3 destination address un-align modulo enable<br>0b - Disable channel 3 destination address un-align modulo<br>1b - Enable channel 3 destination address un-align modulo |
| **5**<br>**SUME2** | RW | Channel 2 source address un-align modulo enable<br>0b - Disable channel 2 source address un-align modulo<br>1b - Enable channel 2 source address un-align modulo |
| **4**<br>**DUME2** | RW | Channel 2 destination address un-align modulo enable<br>0b - Disable channel 2 destination address un-align modulo<br>1b - Enable channel 2 destination address un-align modulo |
| **3**<br>**SUME1** | RW | Channel 1 source address un-align modulo enable<br>0b - Disable channel 1 source address un-align modulo<br>1b - Enable channel 1 source address un-align modulo |
| **2**<br>**DUME1** | RW | Channel 1 destination address un-align modulo enable<br>0b - Disable channel 1 destination address un-align modulo<br>1b - Enable channel 1 destination address un-align modulo |
| **1**<br>**SUME0** | RW | Channel 0 source address un-align modulo enable<br>0b - Disable channel 0 source address un-align modulo<br>1b - Enable channel 0 source address un-align modulo |
| **0**<br>**DUME0** | RW | Channel 0 destination address un-align modulo enable<br>0b - Disable channel 0 destination address un-align modulo<br>1b - Enable channel 0 destination address un-align modulo |

### 17.6.17 DMA_DUMOn (Unalign Modulo N Offset Register)

Offset : 50h+N*4h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SUMO | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DUMO | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_DUMOn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-16**<br>**SUMO** | RW | Lookup table for un-align modulo offset registers of source, can be lookup by CFG_CSRn[13:12]. |
| **15-0**<br>**DUMO** | RW | Lookup table for un-align modulo offset registers of destination, can be lookup by CFG_CSRn[13:12] |

### 17.6.18 DMA_DCHPRI3 (Channel Priority Register)

Offset : 100h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | GRPPRI | | CHPRI | | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**DMA_DCHPRI3 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.19 DMA_DCHPRI2 (Channel Priority Register)

Offset : 101h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | GRPPRI | | CHPRI | | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**DMA_DCHPRI2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |

| | | |
|---|---|---|
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.20 DMA_DCHPRI1 (Channel Priority Register)

Offset : 102h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | GRPPRI | | CHPRI | | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**DMA_DCHPRI1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.21 DMA_DCHPRI0 (Channel Priority Register)

Offset : 103h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | GRPPRI | | CHPRI | | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_DCHPRI0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.22 DMA_DCHPRI7 (Channel Priority Register)

Offset : 104h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | GRPPRI | | CHPRI | | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

**DMA_DCHPRI7 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.23 DMA_DCHPRI6 (Channel Priority Register)

Offset : 105h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | GRPPRI | | CHPRI | | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

**DMA_DCHPRI6 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.24 DMA_DCHPRI5 (Channel Priority Register)

Offset : 106h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | GRPPRI | | CHPRI | | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

**DMA_DCHPRI5 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.25 DMA_DCHPRI4 (Channel Priority Register)

Offset : 107h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | GRPPRI | | CHPRI | | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**DMA_DCHPRI4 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.26 DMA_DCHPRI11 (Channel Priority Register)

Offset : 108h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | GRPPRI | | CHPRI | | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

**DMA_DCHPRI11 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |

| | | |
|---|---|---|
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.27 DMA_DCHPRI10 (Channel Priority Register)

Offset : 109h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | GRPPRI | | CHPRI | | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

**DMA_DCHPRI10 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.28 DMA_DCHPRI9 (Channel Priority Register)

Offset : 10Ah

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | GRPPRI | | CHPRI | | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**DMA_DCHPRI9 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.29 DMA_DCHPRI8 (Channel Priority Register)

Offset : 10Bh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | | GRPPRI | | CHPRI | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

**DMA_DCHPRI8 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.30 DMA_DCHPRI15 (Channel Priority Register)

Offset : 10Ch

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | | GRPPRI | | CHPRI | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**DMA_DCHPRI15 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.31 DMA_DCHPRI14 (Channel Priority Register)

Offset : 10Dh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | | GRPPRI | | CHPRI | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |

**DMA_DCHPRI14 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.32 DMA_DCHPRI13 (Channel Priority Register)

Offset : 10Eh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | GRPPRI | | CHPRI | | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |

**DMA_DCHPRI13 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.33 DMA_DCHPRI12 (Channel Priority Register)

Offset : 10Fh

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | GRPPRI | | CHPRI | | | |
| W | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

**DMA_DCHPRI12 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**GRPPRI** | RW | Channel n Current Group Priority<br>Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read only; writes are ignored. |

| | | |
|---|---|---|
| **3-0**<br>**CHPRI** | RW | Channel n Arbitration Priority<br>Channel priority level for fixed-priority arbitration. |

### 17.6.34 DMA_CFG_SADDRn (CFG Source Address Register)

Offset : 1000h+N*20h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SADDR | | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SADDR | | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DMA_CFG_SADDRn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**SADDR** | RW | Source Address<br>Memory address pointing to the source data. |

### 17.6.35 DMA_CFG_SOFFn (CFG Signed Source Address Offset Register)

Offset : 1004h+N*20h (N : 0 ~ 15)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SOFF | | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DMA_CFG_SOFFn field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**SOFF** | RW | Source Address Signed Offset<br>Sign-extended offset applied to the current source address to form the next-state value as each source read is completed. |

### 17.6.36 DMA_CFG_ATTRn (CFG Transfer Attributes Register)

Offset : 1006h+N*20h (N : 0 ~ 15)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SMOD | | | | | SSIZE | | | DMOD | | | | DSIZE | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DMA_CFG_ATTRn field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-11**<br>**SMOD** | RW | Source Address Modulo<br>00000b - Source address modulo feature is disabled<br>00001-11111b - This value defines a specific address range specified to be the value after SADDR+ SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range. |
| **10-8**<br>**SSIZE** | RW | Source Data Transfer Size<br>000b - 8-bit<br>001b - 16-bit<br>010b - 32-bit<br>011b - 64-bit<br>100b - Reserved<br>101b - 32-byte burst<br>110b - Reserved<br>111b - Reserved |
| **7-3**<br>**DMOD** | RW | Destination Address Modulo<br>See the SMOD definition |
| **2-0**<br>**DSIZE** | RW | Destination Data Transfer Size<br>See the SSIZE definition |

### 17.6.37 DMA_CFG_NBYTES_ILNOn (CFG Inner Byte Count Inner Loop Mapping Disabled Register)

Either one of the three registers DMA_CFG_NBYTES_ILNOn, DMA_CFG_NBYTES_ILOFFNOn, and DMA_CFG_NBYTES_ILOFFYESn defines the number of bytes to be transferred in each request. Those three registers share the same address, and which register will be used depends on the specific configuration of the Inner Loop Mapping.

- If the Inner Loop Mapping is disabled (DMA_CR[EILM] = 0), see the below description of the DMA_CFG_NBYTES_ILNOn register.

Offset : 1008h+N*20h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NBYTES | | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NBYTES | | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DMA_CFG_NBYTES_ILNOn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**NBYTES** | RW | Inner Byte Transfer Count<br>Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate CFG contents load into the DMA engine, and the appropriate reads and writes perform until the inner byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the inner count is done, the SADDR and DADDR values are written back into the CFG memory, the outer loop count is decremented and restored to the CFG memory. If the outer loop count is completed, additional processing is performed. An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer. |

### 17.6.38 DMA_CFG_NBYTES_ILOFFNOn (CFG Signed Inner Loop Offset Inner Loop Mapping Enabled and Offset Disabled Register)

Either one of the three registers DMA_CFG_NBYTES_ILNOn, DMA_CFG_NBYTES_ILOFFNOn, and DMA_CFG_NBYTES_ILOFFYESn defines the number of bytes to be transferred in each request. Those three registers share the same address, and which register will be used depends on the specific configuration of the Inner Loop Mapping.

- If the Inner Loop Mapping is disabled (DMA_CR[EILM] = 1), and the Inner Loop Offset is disabled (SILOE = 0 and DILOE = 0), see the below description of the DMA_CFG_NBYTES_ILOFFNOn register.

Offset : 1008h+N*20h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SILOE | DILOE | NBYTES | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NBYTES | | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DMA_CFG_NBYTES_ILOFFNOn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**SILOE** | RW | Source Inner Loop Offset Enable<br>Selects whether the inner loop offset is applied to the source address upon inner loop completion.<br>0b - The inner loop offset is not applied to the SADDR<br>1b - The inner loop offset is applied to the SADDR |
| **30**<br>**DILOE** | RW | Destination Inner Loop Offset Enable<br>Selects whether the inner loop offset is applied to the destination address upon inner loop completion.<br>0b - The inner loop offset is not applied to the DADDR<br>1b - The inner loop offset is applied to the DADDR |
| **29-0**<br>**NBYTES** | RW | Inner Byte Transfer Count<br>Number of bytes to be transferred in each service request of the channel. |

### 17.6.39 DMA_CFG_NBYTES_ILOFFYESn (CFG Signed Inner Loop Offset Inner Loop Mapping and Offset Enabled Register)

Either one of the three registers DMA_CFG_NBYTES_ILNOn, DMA_CFG_NBYTES_ILOFFNOn, and DMA_CFG_NBYTES_ILOFFYESn defines the number of bytes to be transferred in each request. Those three registers share the same address, and which register will be used depends on the specific configuration of the Inner Loop Mapping.

- If the Inner Loop Mapping is disabled (DMA_CR[EILM] = 1), and the Inner Loop Offset is enabled (SILOE or DILOE = 1), see the below description of the DMA_CFG_NBYTES_ILOFFYESn register.

Offset : 1008h+N*20h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SILOE | DILOE | ILOFF | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ILOFF | | | | | | NBYTES | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DMA_CFG_NBYTES_ILOFFYESn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**SILOE** | RW | Source Inner Loop Offset Enable<br>Selects whether the inner loop offset is applied to the source address upon inner loop completion.<br>0b - The inner loop offset is not applied to the SADDR<br>1b - The inner loop offset is applied to the SADDR |
| **30**<br>**DILOE** | RW | Destination Inner Loop Offset Enable<br>Selects whether the inner loop offset is applied to the destination address upon inner loop completion.<br>0b - The inner loop offset is not applied to the DADDR<br>1b - The inner loop offset is applied to the DADDR |
| **29-10**<br>**ILOFF** | RW | If SILOE or DILOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the inner loop completes. |
| **9-0**<br>**NBYTES** | RW | Inner Byte Transfer Count<br>Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate CFG contents load into the DMA engine, and the appropriate reads and writes perform until the inner byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the inner count is done, the SADDR and DADDR values are written back into the CFG memory, the outer loop count is decremented and restored to the CFG memory. If the outer loop count is completed, additional processing is performed. |

### 17.6.40 DMA_CFG_SLASTn (CFG Last Source Address Adjustment Register)

Offset : 100Ch+N*20h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SLAST | | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SLAST | | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DMA_CFG_SLASTn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**SLAST** | RW | Last Source Address Adjustment<br>Adjustment value added to the source address at the completion of the outer loop count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure.<br>This register uses two's complement notation; the overflow bit is discarded. |

### 17.6.41 DMA_CFG_DADDRn (CFG Destination Address Register)

Offset : 1010h+N*20h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DADDR | | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DADDR | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMA_CFG_DADDRn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**DADDR** | RW | Destination Address<br>Memory address pointing to the destination data. |

### 17.6.42 DMA_CFG_DOFFn (CFG Signed Destination Address Offset Register)

Offset : 1014h+N*20h (N : 0 ~ 15)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DOFF | | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DMA_CFG_DOFFn field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**DOFF** | RW | Destination Address Signed Offset<br>Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed. |

### 17.6.43 DMA_CFG_CLC_CHTRGENNOn (CFG Current Inner Loop Trigger, Outer Loop Count Channel Trigger Disabled Register)

Either one of the two registers DMA_CFG_CLC_CHTRGENNOn, and DMA_CFG_CLC_CHTRGENYESn defines the configuration of the current inner loop trigger and outer loop count. Those two registers share the same address, and which register will be used depends on the specific configuration of the CHTRGEN bit field.

- If CHTRGEN = 0, see the below description of the DMA_CFG_CLC_CHTRGENNOn register.

Offset : 1016h+N*20h (N : 0 ~ 15)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CHTRGEN | CLC | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DMA_CFG_CLC_CHTRGENNOn field descriptions**

| Field | Type | Description |
|---|---|---|
| **15**<br>**CHTRGEN** | RW | Enable Channel-to-Channel Trigger on Inner Loop Completion<br>This bit must be equal to the BLC[CHTRGEN] bit; otherwise, a configuration error is reported.<br>0b - The channel-to-channel trigger is disabled. If channel trigger is disabled, the CLC value is extended to 15 bits in place of a trigger channel number. If the outer loop is exhausted, this trigger mechanism is suppressed in favor of the OCHTREN channel trigger.<br>1b - The channel-to-channel trigger is enabled. As the channel completes the inner loop, this flag enables trigger to another channel, defined by the TRGCH field. The trigger target channel initiates a channel service request via an internal mechanism that sets the CFG_CSRn[START] bit of the specified channel. |

| | | |
|---|---|---|
| **14-0**<br>**CLC** | RW | Current Outer Loop Count<br>This field is the current outer loop count for the channel. It is decremented each time the inner loop is completed and updated in the CFG memory. After the outer loop count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CLC field from the Beginning Loop Count (BLC) field.<br>When the CLC field is initially loaded by software, it must be set to the same value as that contained in the BLC field.<br>If the channel is configured to execute a single service request, the initial values of BLC and CLC should be 0x0001. |

### 17.6.44 DMA_CFG_CLC_CHTRGENYESn (CFG Current Inner Loop Trigger, Outer Loop Count Channel Trigger Enabled Register)

Either one of the two registers DMA_CFG_CLC_CHTRGENNOn, and DMA_CFG_CLC_CHTRGENYESn defines the configuration of the current inner loop trigger and outer loop count. Those two registers share the same address, and which register will be used depends on the specific configuration of the CHTRGEN bit field.

- If CHTRGEN = 1, see the below description of the DMA_CFG_CLC_CHTRGENYESn register.

Offset : 1016h+N*20h (N : 0 ~ 15)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CHTRGEN | 0 | TRGCH | | | | | CLC | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DMA_CFG_CLC_CHTRGENYESn field descriptions**

| Field | Type | Description |
|---|---|---|
| **15**<br>**CHTRGEN** | RW | This bit must be equal to the BLC[CHTRGEN] bit; otherwise, a configuration error is reported.<br>0b - The channel-to-channel trigger is disabled. If channel trigger is disabled, the CLC value is extended to 15 bits in place of a trigger channel number. If the outer loop is exhausted, this trigger mechanism is suppressed in favor of the OCHTREN channel trigger.<br>1b - The channel-to-channel trigger is enabled. Enable channel-to-channel trigger on inner-loop complete<br>As the channel completes the inner loop, this flag enables trigger to another channel, defined by the TRGCH field. The trigger target channel initiates a channel service request via an internal mechanism that sets the CFG_CSRn[START] bit of the specified channel. |

| | | |
|---|---|---|
| **13-9**<br>**TRGCH** | RW | Inner Loop Trigger Channel Number<br>If channel-to-channel trigger is enabled (CHTRGEN = 1), then after the inner loop is done, the DMA engine initiates a channel service request to the channel defined by this field by setting that channel's CFG_CSRn[START] bit. |
| **8-0**<br>**CLC** | RW | Current Outer Loop Count<br>This field is the current outer loop count for the channel. It is decremented each time the inner loop is completed and updated in the CFG memory. After the outer loop count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CLC field from the Beginning Loop Count (BLC) field.<br>When the CLC field is initially loaded by software, it must be set to the same value as that contained in the BLC field.<br>If the channel is configured to execute a single service request, the initial values of BLC and CLC should be 0x0001. |

### 17.6.45 DMA_CFG_DLASTn (CFG Last Destination Address Adjustment Register)

Offset : 1018h+N*20h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DLAST | | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DLAST | | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DMA_CFG_DLASTn field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-0**<br>**DLAST** | RW | DLAST<br>Destination last address adjustment or the memory address Adjustment value added to the destination address at the completion of the outer loop count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure.<br>This field uses two's complement notation for the final destination address adjustment. |

### 17.6.46 DMA_CFG_CSRn (CFG Control and Status Register)

Offset : 101Ch+N*20h (N : 0 ~ 15)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | UMS | OTRGCH | | | | | DONE | ACTIVE | OCHTRGEN | 0 | DREQ | INTHALF | INTOUTER | START |
| W | | | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DMA_CFG_CSRn field descriptions**

| Field | Type | Description |
|---|---|---|
| **13**<br>**UMS** | RW | Un-align Modulo Offset Select<br>Selection of the modulo offset registers.<br>0 - Select Unalign Modulo 0 Offset register for the outer loop trigger channel 0~15; select Unalign Modulo 1 Offset register for the outer loop trigger channel 16~31.<br>1 - Select Unalign Modulo 2 Offset register for the outer loop trigger channel 0~15; select Unalign Modulo 3 Offset register for the outer loop trigger channel 16~31. |
| **12-8**<br>**OTRGCH** | RW | Outer Loop Trigger Channel Number<br>If OCHTRGEN clear, no channel-to-channel trigger, or chaining, is performed after the outer loop counter is exhausted.<br>If OCHTRGEN set, after the outer loop counter is exhausted, the DMA engine initiates a channel service request at the channel defined by this field by setting that channel's CFG_CSRn[START] bit. |
| **7**<br>**DONE** | RW | Channel Done |
| **6**<br>**ACTIVE** | RW | Channel Active<br>This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the DMA as the inner loop completes or when any error condition is detected. |

| | | |
|---|---|---|
| **5**<br>**OCHTRGEN** | RW | Enable Channel-to-Channel Trigger on Outer Loop Completion<br>As the channel completes the outer loop, this flag enables the trigger to another channel, defined by OTRGCH. The trigger target channel initiates a channel service request via an internal mechanism that sets the CFG_CSRn[START] bit of the specified channel.<br>To support the dynamic trigger coherency model, this field is forced to zero when written to while the CFG_CSRn[DONE] bit is set.<br>0b - Disable the channel-to-channel trigger.<br>1b - Enable the channel-to-channel trigger. |
| **3**<br>**DREQ** | RW | Disable Request<br>If this flag is set, the DMA hardware automatically clears the corresponding ERQ bit when the current outer loop count reaches zero.<br>0b - The channel's ERQ bit is not affected.<br>1b - The channel's ERQ bit is cleared when the outer loop is complete. |
| **2**<br>**INTHALF** | RW | Enable an interrupt when outer counter is half complete.<br>If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT register when the current outer loop count reaches the halfway point. Specifically, the comparison performed by the DMA engine is (CLC == (BLC >> 1)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes or other types of data movement where the processor needs an early indication of the transfer's progress. If BLC = 1, do not use INTHALF. Use INTOUTER to instead.<br>0b - Disable the half-point interrupt.<br>1b - Enable the half-point interrupt. |
| **1**<br>**INTOUTER** | RW | Enable an interrupt when outer loop count completes.<br>If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current outer loop count reaches zero.<br>0b - Disable the end-of-outer loop interrupt.<br>1b - Enable the end-of-outer loop interrupt. |
| **0**<br>**START** | RW | Channel Start<br>The DMA hardware automatically clears this flag after the channel begins execution.<br>0b - The channel is not explicitly started.<br>1b - The channel is explicitly started via a software initiated service request, and the channel is requesting service. |

### 17.6.47 DMA_CFG_BLC_CHTRGENNOn (CFG Beginning Inner Loop Trigger, Outer Loop Count Channel Trigger Disabled Register)

Either one of the two registers DMA_CFG_BLC_CHTRGENNOn, and DMA_CFG_BLC_CHTRGENYESn defines the configuration of the beginning inner loop trigger and outer loop count. Those two registers share the same address, and which register will be used depends on the specific configuration of the CHTRGEN bit field.

■ If CHTRGEN = 0, see the below description of the DMA_CFG_BLC_CHTRGENNOn register.

Offset : 101Eh+N*20h (N : 0 ~ 15)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CHTRGEN | BLC | | | | | | | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DMA_CFG_BLC_CHTRGENNOn field descriptions**

| Field | Type | Description |
|---|---|---|
| **15<br>CHTRGEN** | RW | Enable Channel-to-Channel Trigger on Inner Loop Completion<br>As the channel completes the inner loop, this flag enables the trigger to another channel, defined by BLC[TRGCH]. The trigger target channel initiates a channel service request via an internal mechanism that sets the CFG_CSRn[START] bit of the specified channel. If channel trigger is disabled, the BLC value extends to 15 bits in place of a trigger channel number. If the outer loop is exhausted, this trigger mechanism is suppressed in favor of the OCHTRGEN channel trigger.<br>When the software loads the CFG, this field must be set equal to the corresponding CLC field; otherwise, a configuration error is reported. As the outer loop count is exhausted, the contents of this field are reloaded into the CLC field.<br>0b - Disable the channel-to-channel trigger<br>1b - Enable the channel-to-channel trigger |
| **14-0<br>BLC** | RW | Starting Outer Loop Count |

### 17.6.48 DMA_CFG_BLC_CHTRGENYESn (CFG Beginning Inner Loop Trigger, Outer Loop Count Channel Trigger Enabled Register)

Either one of the two registers DMA_CFG_BLC_CHTRGENNOn, and DMA_CFG_BLC_CHTRGENYESn defines the configuration of the beginning inner loop trigger and outer loop count. Those two registers share the same address, and which register will be used depends on the specific configuration of the CHTRGEN bit field.

■ If CHTRGEN = 1, see the below description of the DMA_CFG_BLC_CHTRGENYESn register.

Offset : 101Eh+N*20h (N : 0 ~ 15)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CHTRGEN | 0 | TRGCH | | | | | BLC | | | | | | | | |
| RST | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DMA_CFG_BLC_CHTRGENYESn field descriptions**

| Field | Type | Description |
|---|---|---|
| **15**<br>**CHTRGEN** | RW | Enable Channel-to-Channel Trigger on Inner Loop Completion<br>As the channel completes the inner loop, this flag enables the trigger to another channel, defined by BLC[TRGCH]. The trigger target channel initiates a channel service request via an internal mechanism that sets the CFG_CSRn[START] bit of the specified channel. If channel trigger is disabled, the BLC value extends to 15 bits in place of a trigger channel number. If the outer loop is exhausted, this trigger mechanism is suppressed in favor of the OCHTRGEN channel trigger.<br>When the software loads the CFG, this field must be set equal to the corresponding CLC field; otherwise, a configuration error is reported. As the outer loop count is exhausted, the contents of this field are reloaded into the CLC field.<br>0b - Disable the channel-to-channel trigger<br>1b - Enable the channel-to-channel trigger |
| **13-9**<br>**TRGCH** | RW | Trigger Channel Number<br>If channel-to-channel trigger is enabled (CHTRGEN = 1), then after the inner loop is exhausted, the DMA engine initiates a channel service request at the channel defined by this field by setting that channel's CFG_CSRn[START] bit.<br>When the software loads the CFG, this field must be set equal to the corresponding CLC field; otherwise, a configuration error is reported. As the outer loop count is exhausted, the contents of this field are reloaded into the CLC field. |

| | | |
|---|---|---|
| **8-0**<br>**BLC** | RW | Starting outer loop count<br>As the CFG is first loaded by software, this 9-bit (CHTRGEN = 1) or 15-bit (CHTRGEN = 0) field must be equal to the value in the CLC field. As the outer loop count is exhausted, the contents of this field are reloaded into the CLC field.<br>When the software loads the CFG, this field must be set equal to the corresponding CLC field; otherwise, a configuration error is reported. As the outer loop count is exhausted, the contents of this field are reloaded into the CLC field. If the channel is configured to execute a single service request, the initial values of BLC and CLC should be 0x0001. |

Flagchip

## 17.7 DMAMUX Register Definition

DMAMUX base address : 4000_A000h

**Register Overview - DMAMUX**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h+n*1h** | DMAMUX_CHCFGn (Channel N Configuration Register) | 8 | RW | 00h | page |
| **10h** | DMAMUX_CHTRG (Channel N Trigger Register) | 8 | RW | 00h | page |

### 17.7.1 DMAMUX_CHCFGn (Channel N Configuration Register)

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

Note: Configuring multiple CHCFG registers with the same source value will lead to unpredictable behavior, even if a channel is disabled (ENBL==0). Before changing the trigger or source configurations, a DMA channel must be disabled via CHCFGn[ENBL].

Offset : 0h+N*1h (N : 0 ~ 15)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | ENBL | SOURCE | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMAMUX_CHCFGn field descriptions**

| Field | Type | Description |
|---|---|---|
| **7**<br>**ENBL** | RW | DMA Channel Enable<br>This field is primarily used during configuration of the DMAMUX. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel.<br>0b - Disable DMA channel.<br>1b - Enable DMA channel. |
| **6-0**<br>**SOURCE** | RW | DMA Channel Source (Slot)<br>Select the peripherals and their slot numbers as DMA request source, and routing to a particular DMA channel. |

### 17.7.2 DMAMUX_CHTRG (Channel N Trigger Register)

Offset : 10h

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | | | | TRIG3 | TRIG2 | TRIG1 | TRIG0 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**DMAMUX_CHTRG field descriptions**

| Field | Type | Description |
|---|---|---|
| **3**<br>**TRIG3** | RW | DMA Channel 3 Trigger Enable<br>Enables the periodic trigger capability for the triggered DMA channel 3<br>0b - Disable triggering mode. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode)<br>1b - Enable triggering mode. If triggering is enabled and ENBL is set, the DMAMUX is in Periodic Trigger mode. |
| **2**<br>**TRIG2** | RW | DMA Channel 2 Trigger Enable<br>Enables the periodic trigger capability for the triggered DMA channel 2<br>0b - Disable triggering mode. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode)<br>1b - Enable triggering mode. If triggering is enabled and ENBL is set, the DMAMUX is in Periodic Trigger mode. |
| **1**<br>**TRIG1** | RW | DMA Channel 1 Trigger Enable<br>Enables the periodic trigger capability for the triggered DMA channel 1<br>0b - Disable triggering mode. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode)<br>1b - Enable triggering mode. If triggering is enabled and ENBL is set, the DMAMUX is in Periodic Trigger mode. |
| **0**<br>**TRIG0** | RW | DMA Channel 0 Trigger Enable<br>Enables the periodic trigger capability for the triggered DMA channel 0<br>0b - Disable triggering mode. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode)<br>1b - Enable triggering mode. If triggering is enabled and ENBL is set, the DMAMUX is in Periodic Trigger mode. |

# Chapter 18 Wake-up Unit (WKU)

## 18.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • Updated the wake-up sources for WKU inputs 20, 21, 22 and 23 in Table. *WKU Wake-up sources* in Section *Chip-specific Information;*<br>• Changed the width of the "DELAYTIME" bit field from "3-0" to "2-0" in the register section *WKU_MDC* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 18.2 Chip-specific Information

Using the following internal peripherals as wake-up sources to the Wake-up Unit (WKU) module. WKU_INPUT0 - WKU_INPUT23 are connections to the internal peripheral interrupt sources.

**Table 50. WKU Wake-up sources**

| WKU Input | Wake-up Source |
|---|---|
| WKU_INPUT0 | Reserved |
| WKU_INPUT1 | Reserved |
| WKU_INPUT2 | Reserved |
| WKU_INPUT3 | Reserved |
| WKU_INPUT4 | Reserved |
| WKU_INPUT5 | Reserved |
| WKU_INPUT6 | FCSPI0 |
| WKU_INPUT7 | FCUART |
| WKU_INPUT8 | Reserved |
| WKU_INPUT9 | Reserved |
| WKU_INPUT10 | CMP0 |
| WKU_INPUT11 | CMP1 |
| WKU_INPUT12 | Reserved |
| WKU_INPUT13 | TSTMP0 |
| WKU_INPUT14 | RTC Alarm |
| WKU_INPUT15 | RTC Seconds |
| WKU_INPUT16 | FCPIT0 |
| WKU_INPUT17 | CMU0 |
| WKU_INPUT18 | AONTIMER0 |
| WKU_INPUT19 | GPIOA |
| WKU_INPUT20 | Reserved |
| WKU_INPUT21 | GPIOC |
| WKU_INPUT22 | GPIOD |
| WKU_INPUT23 | GPIOE |

## 18.3 Features

- 24 internal modules with individual control bits for the system to wake up from standby mode.
- Support the wakeup source to trigger a counter.

## 18.4 Overview

The WKU module allows the user to select the internal modules as interrupt wake-up sources from standby mode.

## 18.5 Functional Description

### 18.5.1 WKU Delay Counter

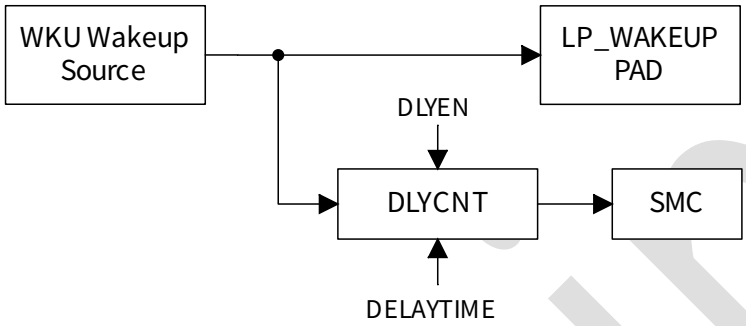
The WKU wakeup source will be sent to SMC to wake up the chip standby mode, and it can be sent to the LP_WAKEUP pad which controlled by CSC0_GPR28 register in the CSC module.

The WKU wakeup source sent to the LP_WAKEUP pad can be optionally delayed by setting the DLYEN register in WKU.

When DLYEN is set, the WKU wakeup source sent to SMC will trigger a counter; meanwhile, the output will be sent to the LP_WAKEUP pad with no delay. The delay time of the WKU wakeup source sent to SMC can be configured by configuring DLYTIME.

When DLYEN is set to 0, the WKU wakeup source will be sent to the LP_WAKEUP pad and SMC at the same time.

**Figure 34. WKU delay counter**




## 18.6 Register Definition

WKU base address : 4003_F000h

**Register Overview - WKU**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **04h** | WKU_MDC (Module Delay Configuration Register0) | 32 | RW | 0000_0000h | page |
| **08h** | WKU_MWER0 (Module Wakeup Enable Register0) | 32 | RW | 0000_0000h | page |
| **0Ch** | WKU_MWER1 (Module Wakeup Enable Register1) | 32 | RW | 0000_0000h | page |
| **10h** | WKU_MWER2 (Module Wakeup Enable Register2) | 32 | RW | 0000_0000h | page |

### 18.6.1 WKU_MDC (Module Delay Configuration Register0)

Offset : 04h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DLYEN | 0 | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | DELAYTIME | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**WKU_MDC field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**DLYEN** | RW | Delay Module Control for Module 31<br>0b - Disable the delay configuration of wakeup source<br>1b - Enable the delay configuration of wakeup source |
| **2-0**<br>**DELAYTIME** | RW | Delay Module Counter Configuration<br>The delay counter clock source is AON_CLK clock<br>The delay time is 2^(DELAYTIME + 3) AON_CLK cycles |

### 18.6.2 WKU_MWER0 (Module Wakeup Enable Register0)

Offset : 08h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MWE7 | MWE6 | MWE5 | MWE4 | MWE3 | MWE2 | MWE1 | MWE0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**WKU_MWER0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **7**<br>**MWE7** | RW | Wakeup Module Control for Module 7<br>0b - Disable WKU_INPUT7 as the wakeup source<br>1b - Enable WKU_INPUT7 as the wakeup source |
| **6**<br>**MWE6** | RW | Wakeup Module Control for Module 6<br>0b - Disable WKU_INPUT6 as the wakeup source<br>1b - Enable WKU_INPUT6 as the wakeup source |
| **5**<br>**MWE5** | RW | Wakeup Module Control for Module 5<br>0b - Disable WKU_INPUT5 as the wakeup source<br>1b - Enable WKU_INPUT5 as the wakeup source |
| **4**<br>**MWE4** | RW | Wakeup Module Control for Module 4<br>0b - Disable WKU_INPUT4 as the wakeup source<br>1b - Enable WKU_INPUT4 as the wakeup source |

| | | |
|---|---|---|
| **3**<br>**MWE3** | RW | Wakeup Module Control for Module 3<br>0b - Disable WKU_INPUT3 as the wakeup source<br>1b - Enable WKU_INPUT3 as the wakeup source |
| **2**<br>**MWE2** | RW | Wakeup Module Control for Module 2<br>0b - Disable WKU_INPUT2 as the wakeup source<br>1b - Enable WKU_INPUT2 as the wakeup source |
| **1**<br>**MWE1** | RW | Wakeup Module Control for Module 1<br>0b - Disable WKU_INPUT1 as the wakeup source<br>1b - Enable WKU_INPUT1 as the wakeup source |
| **0**<br>**MWE0** | RW | Wakeup Module Control for Module 0<br>0b - Disable WKU_INPUT0 as the wakeup source<br>1b - Enable WKU_INPUT0 as the wakeup source |

### 18.6.3 WKU_MWER1 (Module Wakeup Enable Register1)

Offset : 0Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MWE15 | MWE14 | MWE13 | MWE12 | MWE11 | MWE10 | MWE9 | MWE8 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**WKU_MWER1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **7**<br>**MWE15** | RW | Wakeup Module Control for Module 15<br>0b - Disable WKU_INPUT15 as the wakeup source<br>1b - Enable WKU_INPUT15 as the wakeup source |
| **6**<br>**MWE14** | RW | Wakeup Module Control for Module 14<br>0b - Disable WKU_INPUT14 as the wakeup source<br>1b - Enable WKU_INPUT14 as the wakeup source |
| **5**<br>**MWE13** | RW | Wakeup Module Control for Module 13<br>0b - Disable WKU_INPUT13 as the wakeup source<br>1b - Enable WKU_INPUT13 as the wakeup source |

| | | |
|---|---|---|
| **4**<br>**MWE12** | RW | Wakeup Module Control for Module 12<br>0b - Disable WKU_INPUT12 as the wakeup source<br>1b - Enable WKU_INPUT12 as the wakeup source |
| **3**<br>**MWE11** | RW | Wakeup Module Control for Module 11<br>0b - Disable WKU_INPUT11 as the wakeup source<br>1b - Enable WKU_INPUT11 as the wakeup source |
| **2**<br>**MWE10** | RW | Wakeup Module Control for Module 10<br>0b - Disable WKU_INPUT10 as the wakeup source<br>1b - Enable WKU_INPUT10 as the wakeup source |
| **1**<br>**MWE9** | RW | Wakeup Module Control for Module 9<br>0b - Disable WKU_INPUT9 as the wakeup source<br>1b - Enable WKU_INPUT9 as the wakeup source |
| **0**<br>**MWE8** | RW | Wakeup Module Control for Module 8<br>0b - Disable WKU_INPUT8 as the wakeup source<br>1b - Enable WKU_INPUT8 as the wakeup source |

### 18.6.4 WKU_MWER2 (Module Wakeup Enable Register2)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MWE23 | MWE22 | MWE21 | MWE20 | MWE19 | MWE18 | MWE17 | MWE16 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**WKU_MWER2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **7**<br>**MWE23** | RW | Wakeup Module Control for Module 23<br>0b - Disable WKU_INPUT23 as the wakeup source<br>1b - Enable WKU_INPUT23 as the wakeup source |
| **6**<br>**MWE22** | RW | Wakeup Module Control for Module 22<br>0b - Disable WKU_INPUT22 as the wakeup source<br>1b - Enable WKU_INPUT22 as the wakeup source |

| | | |
|---|---|---|
| **5**<br>**MWE21** | RW | Wakeup Module Control for Module 21<br>0b - Disable WKU_INPUT21 as the wakeup source<br>1b - Enable WKU_INPUT21 as the wakeup source |
| **4**<br>**MWE20** | RW | Wakeup Module Control for Module 20<br>0b - Disable WKU_INPUT20 as the wakeup source<br>1b - Enable WKU_INPUT20 as the wakeup source |
| **3**<br>**MWE19** | RW | Wakeup Module Control for Module 19<br>0b - Disable WKU_INPUT19 as the wakeup source<br>1b - Enable WKU_INPUT19 as the wakeup source |
| **2**<br>**MWE18** | RW | Wakeup Module Control for Module 18<br>0b - Disable WKU_INPUT18 as the wakeup source<br>1b - Enable WKU_INPUT18 as the wakeup source |
| **1**<br>**MWE17** | RW | Wakeup Module Control for Module 17<br>0b - Disable WKU_INPUT17 as the wakeup source<br>1b - Enable WKU_INPUT17 as the wakeup source |
| **0**<br>**MWE16** | RW | Wakeup Module Control for Module 16<br>0b - Disable WKU_INPUT16 as the wakeup source<br>1b - Enable WKU_INPUT16 as the wakeup source |

Flagchip

# Chapter 19 Power Management Controller (PMC)

## 19.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 19.2 Features

- Voltage regulators for power supplies on the chip.
- Two power modes.
- Power-on Reset (POR) with brown-out detection.
- Low Voltage Reset (LVR) for all system power domains.
- Low Voltage Detect (LVD) and High Voltage Detect (HVD) as indications for software.

## 19.3 Overview

The Power Management Controller (PMC) controls and provides power supplies for the whole chip. Two 3.3 V or 5 V external inputs are required for normal operation. Voltage regulators, POR generation, and the low/high voltage detect system are integrated to satisfy multiple power options.

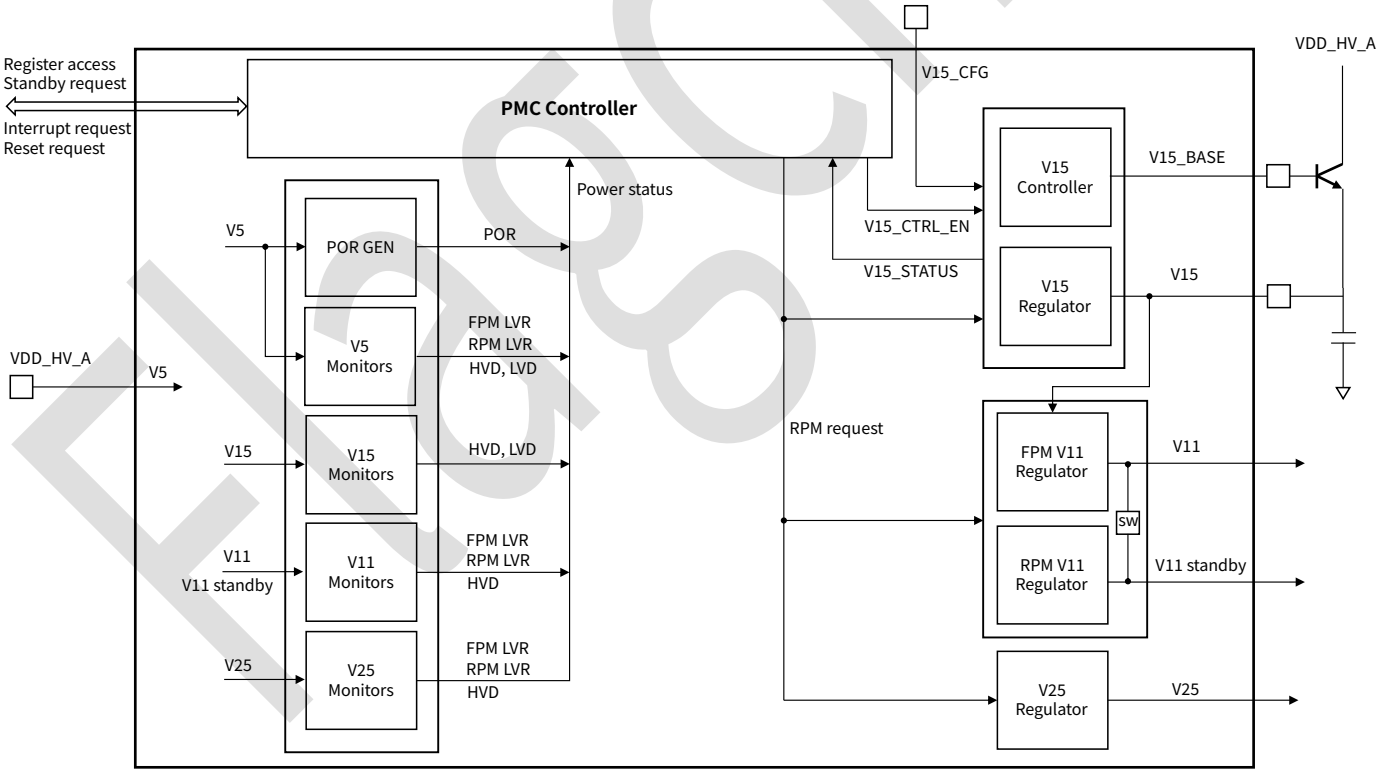
The PMC provides two basic modes of operation for the voltage regulators and monitors:

- Full Performance Mode (FPM): For high-current consumption.
- Reduce Power Mode (RPM): For low-current consumption.

## 19.4 Block Diagram

The following is the block diagram of the PMC module.

**Figure 35. PMC block diagram**




## 19.5 Functional Description

Functional details of the PMC are shown in the following sections.

### 19.5.1 Regulators

Two 3.3 V or 5 V external inputs (V5 or called VDD_HV_A in SOC) are required for normal operation.

The 2.5 V supply (V25) is regulated from V5 by the V25 regulator in the PMC.

The 1.1 V supply (V11) is regulated from V15 by the V11 regulator in the PMC.

The 1.5 V supply (V15) can be generated from V5 by an internal regulator in the PMC or by an internal controller in PMC with an onboard NPN driver, or from an external driver such as an onboard DCDC, depending on the different power requirements. V15_CFG from PAD and V15_CTRL_EN in PMC CONFIG register are used to select the V15 driver source. V15_STATUS in PMC LVSCR register can be used to check which driver source is used to drive V15.

V15_CFG is used to configure whether to use an external driver such as an onboard DCDC or not. If V15_CFG is high, such as pulling up to V5, V15 needs to be driven by an external driver such as an onboard DCDC. If V15_CFG is low, such as tied to the ground, V15 is generated from V5 by an internal regulator in the PMC or by an internal controller in PMC with an onboard NPN driver. Check V15_CTRL_EN in PMC CONFIG register to see how to switch between the internal regulator and controller in the PMC to drive V15.

***Note:*** During power up, V15 can be driven only after V5 is larger than 1.5 V if V15_CFG is high.

### 19.5.2 Resets

The POR and all LVRs in the PMC are combined into one single SOC POR. By reading the POR flag and LVR flags in the PMC_LVSCR register, we can determine which power domain triggered the reset event.

***Note:*** After POR, the user needs to check the LVSCR flag and status bit to see if the MCU powers on successfully.

After the chip's initial power ramps up, the POR flag and the LVR flags are all set to 1. It is required to clear all these flags in the PMC_LVSCR register before usage. So, the root cause can be tracked and debugged in case of an unexpected SOC POR.

### 19.5.3 Interrupts

The PMC generates interrupts as S/W indications of low-voltage or high-voltage events. There are LVD/HVD status bits in PMC LVSCR register which show the status of the power supplies. If the status is changed, an interrupt will be generated:

- HVD interrupt: all HVD monitors are combined into one interrupt source with an Interrupt enable (HVD_IE field in the CONFIG register). See the CONFIG and LVSCR registers for details.
- LVD interrupt: all LVD monitors are combined into one interrupt source with an Interrupt enable (LVD_IE field in the CONFIG register). See the CONFIG and LVSCR registers for details.
- HVD and LVD interrupts are synchronous interrupts, they can only be used in run and wait mode.

Flagchip

## 19.6 Register Definition

PMC base address : 4004_4000h

**Register Overview - PMC**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | PMC_LVSCR (Low Voltage Status and Control Register) | 32 | W1C | 93FF_0030h | page |
| **4h** | PMC_CONFIG (PMC Configuration Register) | 32 | RW | 0000_8000h | page |

### 19.6.1 PMC_LVSCR (Low Voltage Status and Control Register)

Flags or status of system powers.

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | POR_FLAG | 0 | | | V15_STATUS | | 0 | | LVR1P1V_RPM_FLAG | LVR1P1V_FPM_FLAG | LVR2P5V_RPM_FLAG | LVR2P5V_FPM_FLAG | 0 | | LVR5V_RPM_FLAG | LVR5V_FPM_FLAG |
| W | w1c | | | | w1c | | | | w1c | w1c | w1c | w1c | | | w1c | w1c |
| RST | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | HVD1P5V_STATUS | LVD1P5V_STATUS | LVD5V_STATUS | HVD1P1V_STATUS | HVD2P5V_STATUS | 0 | HVD5V_STATUS | 0 | HVD1P5V_FLAG | LVD1P5V_FLAG | LVD5V_FLAG | HVD1P1V_FLAG | HVD2P5V_FLAG | 0 | HVD5V_FLAG |
| W | | | | | | | | | | w1c | w1c | w1c | w1c | w1c | | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |

**PMC_LVSCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**POR_FLAG** | W1C | POR Flag<br>Write 1 to clear (W1C).<br>Can't be cleared by other reset resources.<br>0b - No power-on reset event.<br>1b - Power-on reset event. |

| | | |
|---|---|---|
| **27-26**<br>**V15_STATUS** | W1C | V15 Supply Driver Status<br>00b - V15 is working on internal V15 LDO.<br>01b - V15 is working on internal V15 controller with on board NPN.<br>10b - V15 is driven by external driver such as on board DCDC, V15_CFG PAD is driven high.<br>11b - Reserved. |
| **23**<br>**LVR1P1V_RPM_FLAG** | W1C | Low Voltage Reset Flag of V11 Domain in RPM<br>Write 1 to clear (W1C).<br>Can't be cleared by other reset resources.<br>0b - No low-voltage reset event has occurred.<br>1b - Low-voltage reset event has occurred. |
| **22**<br>**LVR1P1V_FPM_FLAG** | W1C | Low Voltage Reset Flag of V11 Domain in FPM<br>Write 1 to clear (W1C).<br>Can't be cleared by other reset resources.<br>0b - No low-voltage reset event has occurred.<br>1b - Low-voltage reset event has occurred. |
| **21**<br>**LVR2P5V_RPM_FLAG** | W1C | Low Voltage Reset Flag of V25 Domain in RPM<br>Write 1 to clear (W1C).<br>Can't be cleared by other reset resources.<br>0b - No low-voltage reset event has occurred.<br>1b - Low-voltage reset event has occurred. |
| **20**<br>**LVR2P5V_FPM_FLAG** | W1C | Low Voltage Reset Flag of V25 Domain in FPM<br>Write 1 to clear (W1C).<br>Can't be cleared by other reset resources.<br>0b - No low-voltage reset event has occurred.<br>1b - Low-voltage reset event has occurred. |
| **17**<br>**LVR5V_RPM_FLAG** | W1C | Low Voltage Reset Flag of V5 Domain in RPM<br>Write 1 to clear (W1C).<br>Can't be cleared by other reset resources.<br>0b - No low-voltage reset event has occurred.<br>1b - Low-voltage reset event has occurred. |
| **16**<br>**LVR5V_FPM_FLAG** | W1C | Low Voltage Reset Flag of V5 Domain in FPM<br>Write 1 to clear (W1C).<br>Can't be cleared by other reset resources.<br>0b - No low-voltage reset event has occurred.<br>1b - Low-voltage reset event has occurred. |
| **14**<br>**HVD1P5V_STATUS** | RO | HVD1P5V Status on V15 Domain in FPM<br>0b - Voltage on V15 is below high-voltage detect threshold.<br>1b - Voltage on V15 is above high-voltage detect threshold. |

| | | |
|---|---|---|
| **13**<br>**LVD1P5V_STATUS** | RO | LVD1P5V Status on V15 Domain in FPM<br>0b - Voltage on V15 is above low-voltage detect threshold.<br>1b - Voltage on V15 is below low-voltage detect threshold. |
| **12**<br>**LVD5V_STATUS** | RO | LVD5V Status on V5 Domain in FPM<br>0b - Voltage on V5 is above low-voltage detect threshold.<br>1b - Voltage on V5 is below low-voltage detect threshold. |
| **11**<br>**HVD1P1V_STATUS** | RO | HVD11 Status on V11 Domain in FPM<br>0b - Voltage on V11 is below high-voltage detect threshold.<br>1b - Voltage on V11 is above high-voltage detect threshold and FPM. |
| **10**<br>**HVD2P5V_STATUS** | RO | HVD25 Status on V25 Domain in FPM<br>0b - Voltage on V25 is below high-voltage detect threshold.<br>1b - Voltage on V25 is above high-voltage detect threshold and FPM. |
| **8**<br>**HVD5V_STATUS** | RO | HVD5V Status on V5 Domain in FPM<br>0b - Voltage on V5 is below high-voltage detect threshold.<br>1b - Voltage on V5 is above high-voltage detect threshold and FPM. |
| **6**<br>**HVD1P5V_FLAG** | W1C | HVD1P5V Interrupt Flag on V15 Domain in FPM<br>Write 1 to clear (W1C).<br>0b - HVD1P5V_STATUS has not changed.<br>1b - HVD1P5V_STATUS has changed. |
| **5**<br>**LVD1P5V_FLAG** | W1C | LVD1P5V Interrupt Flag on V15 Domain in FPM<br>Write 1 to clear (W1C).<br>0b - LVD1P5V_STATUS has not changed.<br>1b - LVD1P5V_STATUS has changed. |
| **4**<br>**LVD5V_FLAG** | W1C | LVD5V Interrupt Flag on V5 Domain in FPM<br>Write 1 to clear (W1C).<br>0b - LVD5V_STATUS has not changed.<br>1b - LVD5V_STATUS has changed. |
| **3**<br>**HVD1P1V_FLAG** | W1C | HVD1P1V Interrupt Flag on V11 Domain in FPM<br>Write 1 to clear (W1C).<br>0b - HVD1P1V_STATUS has not changed.<br>1b - HVD1P1V_STATUS has changed. |
| **2**<br>**HVD2P5V_FLAG** | W1C | HVD2P5V Interrupt Flag on V25 Domain in FPM<br>Write 1 to clear (W1C).<br>0b - HVD2P5V_STATUS has not changed.<br>1b - HVD2P5V_STATUS has changed. |
| **0**<br>**HVD5V_FLAG** | W1C | HVD5V Interrupt Flag on V5 Domain in FPM<br>Write 1 to clear (W1C).<br>0b - HVD5V_STATUS has not changed.<br>1b - HVD5V_STATUS has changed. |

### 19.6.2 PMC_CONFIG (PMC Configuration Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | V15_LOCK | 0 | | | | | LVD_IE | HVD_IE | 0 | | V15_CTRL_EN | V15_AUTOSW | RPM_VDD2P5_EN | 0 | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PMC_CONFIG field descriptions**

| Field | Type | Description |
|---|---|---|
| **15**<br>**V15_LOCK** | RW | Lock Control for V15_EXT_EN and V15_AUTOSW<br>0b - Unlock, V15_EXT_EN and V15_AUTOSW can be changed.<br>1b - Lock, V15_EXT_EN and V15_AUTOSW cannot be changed. |
| **9**<br>**LVD_IE** | RW | Low Voltage Detect Interrupt Enable<br>0b - LVD interrupt is disabled.<br>1b - LVD interrupt is enabled. |
| **8**<br>**HVD_IE** | RW | High Voltage Detect Interrupt Enable<br>0b - HVD interrupt is disabled.<br>1b - HVD interrupt is enabled. |
| **5**<br>**V15_CTRL_EN** | RW | V15 Controller with Onboard NPN Enable<br>0b - Use internal V15 LDO.<br>1b - Use internal V15 controller with onboard NPN driver. |
| **4**<br>**V15_AUTOSW** | RW | V15 Auto Switch Enable<br>0b - V15 auto switch is disabled.<br>1b - V15 auto switch is enabled, PMC will automatically switch back to V15 LDO driver from V15 controller with on board NPN driver if V15 is low and LVD1P5V event happens, once V15 is stable, it will automatically switch back. |

| | | |
|---|---|---|
| **3**<br>**RPM_VDD2P5_EN** | RW | V25 Domain Enable during RPM<br>0b - V25 regulator are disabled in RPM.<br>1b - V25 regulator are enabled in RPM. |

Flagchip

# Chapter 20 System Mode Controller (SMC)

## 20.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | Added a note for flash DPD and ROM PG to Section *Modes of Operation* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 20.2 Features

### 20.2.1 Introduction

The System Mode Controller (SMC) controls the system's low-power entry and exit sequences. It can function under all low-power modes.

## 20.3 Overview

### 20.3.1 Modes of Operation

The ARM CPU has three primary modes of operation:

- Run
- Sleep
- Deep Sleep

The WFI instruction is used to invoke Sleep and Deep Sleep modes. When core enters Deep Sleep mode, the MCU will enter STOP or STANBY mode according to the setting of the SMC register. The following table shows the translation between the Flagchip MCU power modes and the ARM CPU mode.

**Table 51. Translation between Flagchip MCU mode and ARM CPU mode**

| Flagchip MCU Mode | ARM CPU Mode |
|---|---|
| RUN | Core in RUN mode |
| WAIT | Core in Sleep mode |
| STOP | Core in Deep Sleep mode |
| STANDBY | Core in Deep Sleep mode |

The SMC provides the user with multiple power options. The STANDBY mode can maximum reduce the SoC power by power gating the PD1 domain. The STOP mode can drastically reduce the power while keeping all power on.

Depending on the needs of the user application, a variety of low-power modes are available that allow the state retention, partial power down, or full power down of certain logic and/or memory. I/O states are held in all modes of operation. Several registers are used to configure the various modes of operation for the device.

The following table describes the power modes available for the device.

***Note:***

1) If the flash Deep Power Down (DPD) is used when entering into STOP mode, need to check that flash register MCRS bit 13 is 0 before low-power entry.
2) Access to ROM should be avoided when setting SCM[SYSROM_PGEN].

**Table 52. Available power modes**

| Mode | Description |
|---|---|
| RUN | Core in Run mode and the MCU can run at full speed and the internal supply is fully regulated. |
| WAIT | Core enters sleep mode and no other clock is gated. Only the core clock is gated. |
| STOP | Core enters Deep Sleep mode and both core and bus clocks are gated off. |
| STANDBY | Core enters Deep Sleep mode, all clocks except some low-power clocks can keep on are gated off and the PD1 domain is powered off. There are four variants of standby modes, from Standby0 to Standby3. Differences of the four variants are shown in the below tables. |

**Table 53. Differences in low-power mode behaviors for SRAM in standby mode**

| RAM / Mode | SYS_RAM1(SRAM0, block0,1) | SYS_RAM2(SRAM0, block2,3) | SYS_RAM3(SRAM0,block4,5, ITCM,DTCM) |
|---|---|---|---|
| | 0x2100_0000 - 0x2100_7FFF | 0x2100_8000 - 0x2100_FFFF | 0x2101_0000 - 0x2101_7FFF |
| **STANDBY0** | OFF | OFF | OFF |
| **STANDBY1** | Retention | OFF | OFF |
| **STANDBY2** | OFF | RETENTION | OFF |
| **STANDBY3** | RETENTION | RETENTION | OFF |

## 20.4 Functional Description

### 20.4.1 Power Mode Transitions

The following table defines triggers for various state transitions.

**Table 54. Triggers for state transition**

| Transition | From | To | Trigger Conditions |
|---|---|---|---|
| 1 | RUN | WAIT | Sleep-now or sleep-on-exit modes entered with SLEEPDEEP not set |
| | WAIT | RUN | Interrupt or Reset |
| 2 | RUN | STOP | STOP_MODE field properly configured.<br>Core sleep-now or sleep-on-exit modes entered with SLEEPDEEP set |
| | STOP | RUN | Interrupt or Reset |
| 3 | RUN | STANDBYx | STOP_MODE field properly configured.<br>STANDBY_CFG properly configured.<br>Core sleep-now or sleep-on-exit modes entered with SLEEPDEEP set |
| | STANDBYx | RUN | Interrupt with related WKU register set or Reset or NMI |

***Note:*** If NMI or reset is used as the wakeup source, the NMI or reset assert time should exceed three slow clock cycles to ensure low-power mode can be woken up.

### 20.4.2 Low-Power Entry/Exit Sequence

#### *20.4.2.1 Stop Mode Entry Sequence*

Entry into stop mode is initiated by a CPU executing the WFI instruction. After the instruction is executed, the following sequence occurs:

1) The CPU clock is gated off immediately.
2) Requests are made to all non-CPU bus masters to enter Stop mode.
3) After all masters have acknowledged they are ready to enter Stop mode, requests are made to all bus slaves to enter Stop mode.
4) After all slaves have acknowledged they are ready to enter stop mode, the system and bus clocks are gated off.

#### *20.4.2.2 Stop Mode Exit Sequence*

Exit from a stop mode is initiated either by a reset or an interrupt event. The following sequence is then executed to restore the system to the run mode:

1) System and bus clocks are enabled for all masters and slaves.
2) The CPU clock is enabled and the CPU begins servicing the reset or interrupt that initiated the exit from the stop mode.

#### *20.4.2.3 Standby Mode Entry Sequence*

Entry into standby mode is initiated by a CPU executing the WFI instruction. After the instruction is executed, the following sequence occurs:

1) The core, system, and bus clocks are gated off.
2) Isolation is enabled for PD0 domain.
3) Optionally disable the FIRC/SIRC, and trigger the PMC low-power handshake.
4) PMC turns off the switch, shuts down the PD1 LDO to disable the V11, and optionally disables the V25.

#### *20.4.2.4 Standby Mode Exit Sequence*

Exit from standby mode is initiated either by a reset or an interrupt event. The following sequence is then executed to restore the system to the run mode:

1) Wakeup detected by interrupt/reset event.
2) Wait for PMC FPM ready/FIRC enable ready, and enable the FIRC/SIRC clock source.
3) Reset assert for the PD1 domain.
4) Release isolation of the PD0 domain.
5) Deassert the PD1 domain.

## 20.5 Register Definition

SMC base address : 4004_5000h

**Register Overview - SMC**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **Ch** | SMC_PMCTRL (Power Mode Control Register) | 32 | RW | 0000_0000h | page |
| **10h** | SMC_STANDBY_CFG (Standby Mode Configuration Register) | 32 | RW | 0000_0003h | page |

### 20.5.1 SMC_PMCTRL (Power Mode Control Register)

The PMCTRL register controls entry into Stop or Standby modes. This register is reset on POR and by reset types that trigger POR.

Offset : ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | STOP_MODE | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SMC_PMCTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **2-0**<br>**STOP_MODE** | RW | Stop Mode Control<br>Controls the low power mode type. Any system reset, this bit will be reset to the default value.<br>000b - Stop mode<br>100b - Standby mode<br>Others - Reserved |

### 20.5.2 SMC_STANDBY_CFG (Standby Mode Configuration Register)

The STANDBY_CFG register provides control by user to tune power consumption during the Standby mode selected by the OPTION field. This register is reset on POR and by reset types that trigger POR.

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | OPTION | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**SMC_STANDBY_CFG field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **1-0**<br>**OPTION** | RW | OPTION<br>Standby mode configuration option.<br>00b - Standby0 mode<br>01b - Standby1 mode<br>10b - Standby2 mode<br>11b - Standby3 mode |

Flagchip

# Chapter 21 Reset Generation Module (RGM)

## 21.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | Updated the description in Section *Clock Monitor Unit 3 Reset (CMU)* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 21.2 Features

- Reset source record and reset generation
- Reset filter control
- Reset interrupt control
- Independent reset control for CPU

## 21.3 Overview

The Reset Generation Module (RGM) registers provide reset status information, reset filter control, and reset interrupt control. The RGM registers can be written only in supervisor mode. Write accesses in user mode are blocked and will issue a bus error.

## 21.4 Functional Description

### 21.4.1 Reset Sources

Each reset source has an associated field in the Reset Status (RGM_SRS) register, Sticky Reset Status (SSRS) register, CORE Reset Status (C0_SRS), and Stick CORE Reset Status register (C0_SSRS). The SRS/C0_SRS will record the last reset source and the SSRS/C0_SSRS will record the reset sources since the last POR if related bits are not cleared by software.

#### *21.4.1.1 Power-on Reset (POR)*

When the MCU powers on or when the supply voltage drops below the POR detect voltage, a POR reset occurs. The circuit will hold the MCU in the reset state until the supply has risen above the Low Voltage Detect Reset (LVR) threshold.

#### *21.4.1.2 System Reset*

A system reset comprises the following reset sources.

##### 21.4.1.2.1 System Reset Timeout Reset (SYSRST_TOUT)

When a system reset request happens and the system failed to acknowledge the reset request (for example, a bus clock loss), a system reset timeout reset will happen after about 2 ms. And the SRS[SYSRST_TOUT] will assert to indicate this reset source happens.

##### 21.4.1.2.2 Pin Reset Timeout Reset (PINRST_TOUT)

When a reset pin is pulled low and a reset does not happen, a pin reset timeout reset will happen after about 2 ms. And the SRS[PINRST_TOUT] and SRS[SYSRST_TOUT] will assert to indicate this reset source happens.

##### 21.4.1.2.3 Reset FSM Error Reset (FSM_ERR)

When the reset control FSM transfers to an error state or error reset sequence happens, an FSM error reset will happen and the SRS[FSM_ERR] will assert to indicate this reset source happens.

##### 21.4.1.2.4 Logic BIST Reset (LBIST)

A logic BIST reset can be generated by writing to STCU-related registers. When a logic BIST reset happens and the SRS[LBIST] will assert to indicate this reset source happens.

##### 21.4.1.2.5 Clock Monitor Unit 3 Reset (CMU)

A CMU3 reset will be generated by an SIRC error (mismatch or loss of clock). An SIRC error is enabled when the CMU3 is enabled and configurated to generate a reset on errors (by writing 1 to CSC0_CMU_CTRL[CMU3_RST_EN] field, and if CSC0_CMU_CTRL[CMU3_FCSMU_RST] is set, CMU3 will reset FCSMU, too).

##### 21.4.1.2.6 Stop Mode Acknowledge Error (SACKERR)

When the MCU attempts to enter low power mode but not all modules acknowledge a stop mode request in the required time of approximately 512 ms, a stop acknowledge error reset will be generated. And the SRS[SACKERR] will assert to

indicate this reset source happens. The timeout counter is the AON32K clock. See AONCLKSR[AON32KCLKSEL] register for details on the AON32K clock selection.

#### 21.4.1.2.7 SYS-AP System Reset Request (SYSAP)

A system reset can be issued by writing 1 to the system reset request field in the SYS-AP control register. This is also the method for resets via the JTAG/SWD interface. The system reset is held until this field is written to 0. And the SRS[SYSAP] will assert to indicate this reset source happens.

#### 21.4.1.2.8 JTAG Reset (JTAG)

The JTAG module generates a system reset when certain IR codes are selected. This functional reset is asserted when EXTEST, HIGHZ, and CLAMP instructions are active. The reset source from the JTAG module is released when any other IR code is selected. A JTAG reset causes the MCU to write 1 to the RGM_SRS[JTAG] field.

#### 21.4.1.2.9 Pin Reset (PIN)

When the reset pad pulls low, the MCU will reset, and this can also wake up the MCU from any low power mode. If used as a reset pin, make sure the pad is configured as the reset pin function. When the MCU is in the reset state, the reset pad will internally drive low, and after the reset sequence is done, it automatically goes to high. When a pin reset happens, the RGM_SRS[PIN] and RGM_SSRS[PIN] will assert.

#### 21.4.1.2.10 HSM Watchdog Reset (HSM_WDOG)

The HSM Watchdog timer (HSM_WDOG) monitors the operation of the FlexCore by expecting periodic refreshing of the watchdog by software. If a watchdog timeout happens, the watchdog issues an HSM system reset. And if the HSM watchdog is configured as a system reset, the RGM_SRS[HSM_WDOG] will assert after the watchdog reset happens.

#### *21.4.1.2.11* Fault Control and Safety Management Unit (FCSMU) Reset

The FCSMU will generate a reset when a hardware recoverable fault occurs, such as a Safety Test and Control Unit (STCU) fault, Low Voltage Detect (LVD) fault, and so on. To enable this feature, it is needed to configure specific channels of the FCSMU as reset mode. For detailed FCSMU configuration steps, refer to the FCSMU chapter.

#### 21.4.1.2.12 Clock Error0 (CLKERR0) Reset

A CLKERR0 reset is enabled when a PLL0/PLL1/FOSC/SOSC clock monitor is enabled and configured to generate a reset on errors (by writing 1 to both SCG_PLL0CSR[CM] and SCG_PLL0CSR[CMRE]). CLKERR0 comprises all PLL errors, including PLL loss of clock, PLL loss of lock, and FOSC loss of clock (if FOSC is selected as the PLL reference clock). After the CLKERR0 reset happens, the RGM_SRS[CLKERR0] asserts.

#### 21.4.1.2.13 Clock Error1 (CLKERR1) Reset

A CLKERR1 reset will be generated by an internal or external system clock error and an SCG state error. Internal system clock errors include PLL0 loss of clock (if PLL0 is selected as the system clock and PLL0 clock monitor is enabled), FOSC loss of clock (if FOSC is selected as the system clock and FOSC clock monitor is enabled), and slow clock loss of clock (if SCG_CCR[SYSCLK_CME] is enabled). The external system clock error is generated by CMU4 to monitor the slow clock. An SCG state error will happen if an SCG internal WDOG error happens when entering or exiting low power mode. After the CLKERR1 reset, the RGM_SRS[CLKERR1] asserts.

#### 21.4.1.2.14 Low Voltage Detect Reset (LVR)

The LVR system will generate a reset when a low voltage condition is detected. After an LVR reset occurs, the MCU will remain in reset until the supply voltage has risen above the LVR threshold voltage. After the LVR reset or POR reset, the RGM_SRS[LVR] asserts.

#### *21.4.1.3 Core Related Reset*

The below reset sources are CORE-related reset sources.

##### 21.4.1.3.1 Interrupt Monitor Timeout Reset (INTM_TOUT)

The Interrupt Monitor (INTM) module generates a system reset when the peripheral's interrupt is not serviced in the configured time. The C0_SRS[C0_INTM] will assert after an interrupt monitor timeout reset.

##### 21.4.1.3.2 System Core Reset (SYSRST)

The SYSRESETREQ bit in the AIRCR register of the ARM core can be written to 1 to issue a software reset (See ARM's documentation for a full description of the register fields). After the software reset, the C0_SRS[C0_SYSRST] field will assert.

##### 21.4.1.3.3 Lockup Reset (LOCKUP)

When a LOCKUP event happens in the core, the MCU will issue a reset (See ARM Cortex-M7 Devices Generic User Guide for details on a "lockup" event). After the lockup reset happens, the C0_SRS[C0_LOCKUP] will assert.

##### 21.4.1.3.4 System Software Reset (SWRST)

A core reset can be generated by writing to RGM C0_RST register. Writing to C0_RST register will reset the related core, and after reset, the C0_SRS[C0_SWRST] will assert to indicate a system software reset happens.

##### 21.4.1.3.5 Watchdog Reset (WDOG)

The watchdog timer (WDOG0) monitors the operation of the system by expecting periodic refreshing of the watchdog by software. If a watchdog timeout happens, the watchdog issues a system reset. After the watchdog reset happens, the C0_SRS[C0_WDOG] will assert according to the timeout watchdog.

### 21.4.2 Reset Interrupt

The RGM can generate two types of interrupts:

- The RGM supports generating an interrupt before a system reset takes effect, and the delay between the interrupt and reset is configurable. This gives a way for software to perform some operations before resetting. ***Note:*** Not all reset sources support this kind of reset interrupt, please check related registers for details.

### 21.4.3 Reset Pin Filter

The reset pin filter function can be used to filter signal glitches on the reset pad. The reset filter can use a bus clock or AON32K clock, but in low power mode, only the AON32K clock can be used for filter. When the bus clock is used as a filter clock, the RST[RSTFLT_BUSW] register can configure the reset pin filter clock number. When the AON32K clock is used as a reset filter clock, the reset signal whose length is less than two periods of the AON32K clock will be filtered. See CSC0_AONCLKSR [AON32KCLKSEL] register for details on the AON32K clock selection.

## 21.5 Register Definition

RGM base address : 4004_6000h

**Register Overview - RGM**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **8h** | RGM_SRS (System Reset Status Register) | 32 | RO | 0000_0082h | page |
| **Ch** | RGM_RSTFLT (Reset Filter Control Register) | 32 | RW | 0000_0000h | page |
| **18h** | RGM_SSRS (Sticky System Reset Status Register) | 32 | RW | 0000_0082h | page |
| **1Ch** | RGM_SRIE (System Reset Interrupt Enable Register) | 32 | RW | 0000_0000h | page |
| **100h** | RGM_C0_CFG (CPU0 Reset Configuration Register) | 32 | RW | 001F_0000h | page |
| **104h** | RGM_C0_RST (CPU0 Reset Register) | 32 | RW | 0000_0000h | page |
| **108h** | RGM_C0_SRS (CPU0 System Reset Status Register) | 32 | RW | 0000_0082h | page |
| **114h** | RGM_C0_SSRS (CPU0 Sticky System Reset Status Register) | 32 | RW | 0000_0082h | page |

### 21.5.1 RGM_SRS (System Reset Status Register)

The read-only status flags in this register indicate the source of the last reset. If multiple reset events happen at the same time, multiple flags can be set. The reset value of this register depends on the reset source. A POR reset will assert LVR and POR bit field in the related status register. An LVR reset without POR will only assert the LVR bit field.

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SYSRST_TOUT | PINRST_TOUT | FSM_ERR | 0 | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LBIST | CMU | SACKERR | WDOG1 | SYSAP | 0 | | JTAG | POR | PIN | HSM_WDOG | FCSMU | CLKERR0 | CLKERR1 | LVR | WAKEUP |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**RGM_SRS field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31**<br>**SYSRST_TOUT** | RO | System Reset Request Timeout Reset<br>0b - Reset source is not a system reset request timeout<br>1b - Reset source is a system reset request timeout |
| **30**<br>**PINRST_TOUT** | RO | Pin Reset Request Timeout Reset<br>0b - Reset source is not pin reset request timeout<br>1b - Reset source is pin reset request timeout |
| **29**<br>**FSM_ERR** | RO | FSM Error Reset<br>0b - Reset source is not an FSM error<br>1b - Reset source is an FSM error |
| **15**<br>**LBIST** | RO | Logic BIST Reset<br>0b - Reset source is not logic BIST<br>1b - Reset source is logic BIST |
| **14**<br>**CMU** | RO | CMU Reset<br>0b - Reset source is not CMU<br>1b - Reset source is CMU |
| **13**<br>**SACKERR** | RO | Stop Acknowledge Error<br>Reset caused by peripheral stop acknowledge timeout during low power mode entry.<br>0b - Reset source is not a peripheral stop acknowledge error<br>1b - Reset source is a peripheral stop acknowledge error |
| **12**<br>**WDOG1** | RO | WDOG1 Reset<br>0b - Reset source is not WDOG1<br>1b - Reset source is WDOG1 |
| **11**<br>**SYSAP** | RO | SYSAP System Reset<br>Reset caused by SYSAP system reset bit written.<br>0b - Reset source is not SYSAP<br>1b - Reset source is SYSAP |
| **8**<br>**JTAG** | RO | JTAG Generated Reset<br>Reset caused by JTAG.<br>0b - Reset source is not JTAG<br>1b - Reset source is JTAG |
| **7**<br>**POR** | RO | Power-On Reset<br>Reset caused by power on reset.<br>0b - Reset source is not power on reset<br>1b - Reset source is power on reset |
| **6**<br>**PIN** | RO | Pin Reset<br>Reset caused by reset pin.<br>0b - Reset source is not reset pin<br>1b - Reset source is reset pin |

| | | |
|---|---|---|
| **5**<br>**HSM_WDOG** | RO | HSM WDOG Reset<br>0b - Reset source is not HSM WDOG<br>1b - Reset source is HSM WDOG |
| **4**<br>**FCSMU** | RO | FCSMU Reset<br>Reset caused by an FCSMU error.<br>0b - Reset source is not an FCSMU error<br>1b - Reset source is an FCSMU error |
| **3**<br>**CLKERR0** | RO | Clock Error0 Reset<br>Reset caused by Clock Error0.<br>0b - Reset source is not Clock Error0.<br>1b - Reset source is Clock Error0. |
| **2**<br>**CLKERR1** | RO | Clock Error1 Reset<br>Reset caused by Clock Error1.<br>0b - Reset source is not Clock Error1.<br>1b - Reset source is Clock Error1. |
| **1**<br>**LVR** | RO | Low Voltage Detect Reset<br>Reset caused by LVR<br>0b - Reset source is not an LVR or a POR<br>1b - Reset source is an LVR or a POR |
| **0**<br>**WAKEUP** | RO | WAKEUP Reset<br>Reset caused by wakeup from standby<br>0b - Reset source is not low power wakeup<br>1b - Reset source is low power wakeup |

### 21.5.2 RGM_RSTFLT (Reset Filter Control Register)

RSTFLT register provides the function to filter the glitch on the reset pad. This register is reset on POR only, other reset types will have no effect.

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | RSTFLT_BUSW | | | | | 0 | | | | | RSTFLT_AON_LP | RSTFLT_AON | RSTFLT_BUS |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RGM_RSTFLT field descriptions**

| Field | Type | Description |
|---|---|---|
| **12-8**<br>**RSTFLT_BUSW** | RW | Reset Pin Filter Bus Clock Width Configuration<br>Configure the number of bus clocks of reset pin filter<br>・ Reset signals whose lengths are less than RSTFLTSEL cycles are filtered.<br>・ Reset signals whose lengths are between RSTFLTSEL and (RSTFLTSEL+1) cycles may be filtered.<br>・ Reset signals whose lengths are greater than (RSTFLTSEL+1) cycles are not filtered. |
| **2**<br>**RSTFLT_AON_LP** | RW | Reset Pin Filter Using AON32K Clock in Low Power Mode<br>The reset pin filter enable configuration in low power mode using AON32K clock. A reset signal whose length is less than 2 AON32K clock periods will be filtered. For 32K clock configuration see CSC0 register AONCLKSEL.<br>0b - Filter function disabled<br>1b - Filter function enabled |
| **1**<br>**RSTFLT_AON** | RW | Reset Pin Filter ENABLE Using AON32K Clock<br>The reset pin filter enable configuration in run mode using AON32K clock. A reset signal whose length is less than 2 AON32K clock periods will be filtered.<br>0b - Filter function disabled<br>1b - Filter function enabled |
| **0**<br>**RSTFLT_BUS** | RW | Reset Pin Filter ENABLE Using Bus Clock<br>The reset pin filter enable configuration in run mode using bus clock.<br>0b - Filter function disabled<br>1b - Filter function enabled |

### 21.5.3 RGM_SSRS (Sticky System Reset Status Register)

Status Flags in this register indicate all reset sources since the last POR or LVR which not been cleared by software. Flag bits in this register are write one to clear type.

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SYSRST_TOUT | PINRST_TOUT | FSM_ERR | 0 | | | | | | | | | | | | |
| W | w1c | w1c | w1c | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LBIST | CMU | SACKERR_ST | WDOG1 | SYSAP_ST | 0 | | JTAG_ST | POR_ST | PIN_ST | HSMWDOG_ST | FCSMU_ST | CLKERR0_ST | CLKERR1_ST | LVR | WAKEUP_ST |
| W | w1c | w1c | w1c | w1c | w1c | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**RGM_SSRS field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**SYSRST_TOUT** | W1C | System Reset Request Timeout Reset<br>0b - Reset source is not a system reset request timeout<br>1b - Reset source is a system reset request timeout |
| **30**<br>**PINRST_TOUT** | W1C | Pin Reset Request Timeout Reset<br>0b - Reset source is not a pin reset request timeout<br>1b - Reset source is a pin reset request timeout |
| **29**<br>**FSM_ERR** | W1C | FSM Error Reset<br>0b - Reset source is not an FSM error<br>1b - Reset source is an FSM error |
| **15**<br>**LBIST** | W1C | Logic BIST Reset<br>0b - Reset source is not logic BIST<br>1b - Reset source is logic BIST |
| **14**<br>**CMU** | W1C | CMU Reset<br>0b - Reset source is not CMU<br>1b - Reset source is CMU |
| **13**<br>**SACKERR_ST** | W1C | Sticky Stop Acknowledge Error<br>Reset caused by peripheral stop acknowledge timeout during low power mode entry.<br>0b - Reset source is not a peripheral stop acknowledge error<br>1b - Reset source is a peripheral stop acknowledge error |
| **12**<br>**WDOG1** | W1C | WDOG1 Reset<br>0b - Reset source is not WDOG1<br>1b - Reset source is WDOG1 |

| | | |
|---|---|---|
| **11**<br>**SYSAP_ST** | W1C | Sticky SYSAP System Reset<br>Reset caused by SYSAP system reset bit written.<br>0b - Reset source is not SYSAP<br>1b - Reset source is SYSAP |
| **8**<br>**JTAG_ST** | W1C | Sticky JTAG generated reset<br>Reset caused by JTAG.<br>0b - Reset source is not JTAG<br>1b - Reset source is JTAG |
| **7**<br>**POR_ST** | W1C | Sticky Power-On Reset<br>Reset caused by power on reset.<br>0b - Reset source is not power on reset<br>1b - Reset source is power on reset |
| **6**<br>**PIN_ST** | W1C | Sticky Pin Reset<br>Reset caused by reset pin.<br>0b - Reset source is not reset pin<br>1b - Reset source is reset pin |
| **5**<br>**HSMWDOG_ST** | W1C | Sticky HSM Watchdog Reset<br>Reset caused by the HSM watchdog timeout.<br>0b - Reset source is not HSM watchdog timeout<br>1b - Reset source is HSM watchdog timeout |
| **4**<br>**FCSMU_ST** | W1C | Sticky FCSMU Reset<br>Reset caused by the FCSMU.<br>0b - Reset source is not FCSMU<br>1b - Reset source is FCSMU |
| **3**<br>**CLKERR0_ST** | W1C | Sticky Clock Error0 Reset<br>Reset caused by Clock Error0.<br>0b - Reset source is not Clock Error0.<br>1b - Reset source is Clock Error0. |
| **2**<br>**CLKERR1_ST** | W1C | Sticky Clock Error1 Reset<br>Reset caused by Clock Error1.<br>0b - Reset source is not Clock Error1.<br>1b - Reset source is Clock Error1. |
| **1**<br>**LVR** | W1C | Low Voltage Detect Reset<br>0b - Reset source is not an LVR or a POR<br>1b - Reset source is an LVR or a POR |
| **0**<br>**WAKEUP_ST** | W1C | Sticky WAKEUP Reset<br>Reset caused by wakeup from standby.<br>0b - Reset source is not low power wakeup<br>1b - Reset source is low power wakeup |

### 21.5.4 RGM_SRIE (System Reset Interrupt Enable Register)

This register can delay the assertion of a system reset and generate an interrupt first. In this way, software can perform some procedure before reset. Notice that a chip POR reset cannot be delayed by this feature. This feature uses AON32K clock to count the delay. This register is reset on Chip POR only.

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | SACKERR_RIE | WDOG1_RIE | SYSAP_RIE | SW_RIE | CPULOC_RIE | JTAG_RIE | GLOBAL_RIE | PIN_RIE | WDG_RIE | 0 | CLKERR0_RIE | 0 | DELAY | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RGM_SRIE field descriptions**

| Field | Type | Description |
|---|---|---|
| **13**<br>**SACKERR_RIE** | RW | Stop Acknowledge Error Reset Interrupt Enable<br>0b - SACKERR reset interrupt disable<br>1b - SACKERR reset interrupt enable |
| **12**<br>**WDOG1_RIE** | RW | WDOG1 Reset Interrupt Enable<br>0b - WDOG1 reset interrupt disable<br>1b - WDOG1 interrupt enable |
| **11**<br>**SYSAP_RIE** | RW | SYSAP System Reset Interrupt Enable<br>0b - SYSAP reset interrupt disable<br>1b - SYSAP reset interrupt enable |
| **10**<br>**SW_RIE** | RW | Software Reset Interrupt Enable<br>0b - Software reset interrupt disable<br>1b - Software reset interrupt enable |
| **9**<br>**CPULOC_RIE** | RW | Core Lockup Reset Interrupt Enable<br>0b - Core LOCKUP reset interrupt enable<br>1b - Core LOCKUP reset interrupt disable |
| **8**<br>**JTAG_RIE** | RW | JTAG Generated Reset Interrupt Enable<br>0b - JTAG reset interrupt disable<br>1b - JTAG reset interrupt enable |

| | | |
|---|---|---|
| **7**<br>**GLOBAL_RIE** | RW | Global Reset Interrupt Enable<br>This field must be set to 1 if the reset delay function needs to be used.<br>0b - Global reset interrupt disable<br>1b - Global reset interrupt enable |
| **6**<br>**PIN_RIE** | RW | PIN Reset Interrupt Enable<br>0b - PIN reset interrupt disable<br>1b - PIN reset interrupt enable |
| **5**<br>**WDG_RIE** | RW | Watchdog Reset Interrupt Enable<br>0b - Watchdog reset interrupt disable<br>1b - Watchdog reset interrupt enable |
| **3**<br>**CLKERR0_RIE** | RW | Clock Error0 Reset Interrupt Enable<br>0b - CLKERR0 reset interrupt disable<br>1b - CLKERR0 reset interrupt enable |
| **1-0**<br>**DELAY** | RW | Reset Delay<br>Configures the reset delay time from when the interrupt asserts and the reset eventually occurs (actually time needs to plus 2 clocks for sync). The clock used is 32K clock which is selected by CSC0_AONCLKSEL[RTCCLKSEL].<br>00b - 8 clock cycles<br>01b - 32 clock cycles<br>10b - 128 clock cycles<br>11b - 512 clock cycles |

### 21.5.5 RGM_C0_CFG (CPU0 Reset Configuration Register)

This register can configure CPU0 related interrupt enable.

Offset : 100h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | C0_SWRST_EN | C0_INTM_EN | C0_WDOG_EN | C0_SYSRST_EN | C0_LOCKUP_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | C0_SWRST_IE | C0_INTM_IE | C0_WDOG_IE | C0_SYSRST_IE | C0_LOCKUP_IE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RGM_C0_CFG field descriptions**

| Field | Type | Description |
|---|---|---|
| **20**<br>**C0_SWRST_EN** | RO | CPU0 SW Reset Enable<br>CPU0 SW reset will always trigger a system reset. This bit is always read as 1'b1.<br>0b - Disable CPU0 SW reset<br>1b - Enable CPU0 SW reset |
| **19**<br>**C0_INTM_EN** | RO | CPU0 INTM Reset Enable<br>CPU0 INTM reset will always trigger a system reset. This bit is always read as 1'b1.<br>0b - Disable CPU0 INTM reset<br>1b - Enable CPU0 INTM reset |
| **18**<br>**C0_WDOG_EN** | RO | CPU0 WDOG Reset Enable<br>CPU0 WDOG reset will always trigger a system reset. This bit is always read as 1'b1.<br>0b - Disable CPU0 WDOG reset<br>1b - Enable CPU0 WDOG reset |
| **17**<br>**C0_SYSRST_EN** | RO | CPU0 SYSRST Reset Enable<br>CPU0 SYSRST reset will always trigger a system reset. This bit is always read as 1'b1.<br>0b - Disable CPU0 SYSRST reset<br>1b - Enable CPU0 SYSRST reset |
| **16**<br>**C0_LOCKUP_EN** | RO | CPU0 LOCKUP Reset Enable<br>CPU0 LOCKUP reset will always trigger a system reset. This bit is always read as 1'b1.<br>0b - Disable CPU0 LOCKUP reset<br>1b - Enable CPU0 LOCKUP reset |
| **4**<br>**C0_SWRST_IE** | RW | CPU0 SWRST Interrupt Enable<br>0b - Disable CPU0 SWRST interrupt<br>1b - Enable CPU0 SWRST interrupt |
| **3**<br>**C0_INTM_IE** | RW | CPU0 INTM Interrupt Enable<br>0b - Disable CPU0 INTM interrupt<br>1b - Enable CPU0 INTM interrupt |
| **2**<br>**C0_WDOG_IE** | RW | CPU0 WDOG Interrupt Enable<br>0b - Disable CPU0 WDOG interrupt<br>1b - Enable CPU0 WDOG interrupt |

| | | |
|---|---|---|
| **1**<br>**C0_SYSRST_IE** | RW | CPU0 SYSRST Interrupt Enable<br>0b - Disable CPU0 SYSRST interrupt<br>1b - Enable CPU0 SYSRST interrupt |
| **0**<br>**C0_LOCKUP_IE** | RW | CPU0 LOCKUP Interrupt Enable<br>0b - Disable CPU0 LOCKUP interrupt<br>1b - Enable CPU0 LOCKUP interrupt |

### 21.5.6 RGM_C0_RST (CPU0 Reset Register)

CPU0 software reset control and status register.

Offset : 104h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | C0_OUT_OF_RST | 0 |
| W | | | | | | | | | | | | | | | | C0_SWRST |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RGM_C0_RST field descriptions**

| Field | Type | Description |
|---|---|---|
| **1**<br>**C0_OUT_OF_RST** | RO | CPU0 Out of Reset Flag<br>0b - CPU0 is under reset<br>1b - CPU0 is out of reset |
| **0**<br>**C0_SWRST** | WO | CPU0 SW Reset Control<br>Writing 1 to this field will issue a CPU0 reset. |

### 21.5.7 RGM_C0_SRS (CPU0 System Reset Status Register)

CPU0 system reset status register. This register will provide the last reset source.

Offset : 108h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SYSRST_TOUT | PINRST_TOUT | FSM_ERR | 0 | | | | | | | | C0_SWRST | C0_INTM | C0_WDOG | C0_SYSRST | C0_LOCKUP |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | C0_SRS | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**RGM_C0_SRS field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**SYSRST_TOUT** | RO | System Reset Request Timeout Reset<br>0b - Reset source is not a system reset request timeout<br>1b - Reset source is a system reset request timeout |
| **30**<br>**PINRST_TOUT** | RO | Pin Reset Request Timeout Reset<br>0b - Reset source is not pin reset request timeout<br>1b - Reset source is pin reset request timeout |
| **29**<br>**FSM_ERR** | RO | FSM Error Reset<br>0b - Reset source is not an FSM error<br>1b - Reset source is an FSM error |
| **20**<br>**C0_SWRST** | RO | CPU0 SW Reset<br>0b - Reset source is not CPU0 SWRST<br>1b - Reset source is CPU0 SWRST |
| **19**<br>**C0_INTM** | RO | CPU0 INTM Reset<br>0b - Reset source is not CPU0 INTM<br>1b - Reset source is CPU0 INTM |
| **18**<br>**C0_WDOG** | RO | CPU0 WDOG Reset<br>0b - Reset source is not CPU0 WDOG<br>1b - Reset source is CPU0 WDOG |
| **17**<br>**C0_SYSRST** | RO | CPU0 SYSRST Reset<br>0b - Reset source is not CPU0 SYSRST<br>1b - Reset source is CPU0 SYSRST |

| | | |
|---|---|---|
| **16**<br>**C0_LOCKUP** | RO | CPU0 LOCKUP Reset<br>0b - Reset source is not CPU0 LOCKUP<br>1b - Reset source is CPU0 LOCKUP |
| **15-0**<br>**C0_SRS** | RO | See description of RGM_SRS |

### 21.5.8 RGM_C0_SSRS (CPU0 Sticky System Reset Status Register)

CPU0 system reset status stick register. This register will provide reset sources since the last POR.

Offset : 114h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SYSRST_TOUT | PINRST_TOUT | FSM_ERR | 0 | | | | | | | | C0_SWRST | C0_INTM | C0_WDOG | C0_SYSRST | C0_LOCKUP |
| W | w1c | w1c | w1c | | | | | | | | | w1c | w1c | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SSRS | | | | | | | | | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**RGM_C0_SSRS field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**SYSRST_TOUT** | W1C | System Reset Timeout<br>0b - Reset source is not a system reset timeout<br>1b - Reset source is a system reset timeout |
| **30**<br>**PINRST_TOUT** | W1C | Pin Reset Timeout<br>0b - Reset source is not a pin reset timeout<br>1b - Reset source is a pin reset timeout |
| **29**<br>**FSM_ERR** | W1C | FSM Error Reset<br>0b - Reset source is not an FSM error<br>1b - Reset source is an FSM error |
| **20**<br>**C0_SWRST** | W1C | CPU0 SW Reset<br>0b - Reset source is not CPU0 SWRST<br>1b - Reset source is CPU0 SWRST |

| | | |
|---|---|---|
| **19**<br>**C0_INTM** | W1C | CPU0 INTM Reset<br>0b - Reset source is not CPU0 INTM<br>1b - Reset source is CPU0 INTM |
| **18**<br>**C0_WDOG** | W1C | CPU0 WDOG Reset<br>0b - Reset source is not CPU0 WDOG<br>1b - Reset source is CPU0 WDOG |
| **17**<br>**C0_SYSRST** | W1C | CPU0 SYSRST Reset<br>0b - Reset source is not CPU0 SYSRST<br>1b - Reset source is CPU0 SYSRST |
| **16**<br>**C0_LOCKUP** | W1C | CPU0 LOCKUP Reset<br>0b - Reset source is not CPU0 LOCKUP<br>1b - Reset source is CPU0 LOCKUP |
| **15-0**<br>**SSRS** | W1C | See description of RGM_SSRS |

Flagchip

# Chapter 22 Security Controller (SEC)

## 22.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • Added a new bit field "BOOT_SUCCESS[18]" to the register section *SEC_BRC0;*<br>• Updated the description of the "FLEXCORE_EN" bit field in the register section *SEC_FLEXCORE_EN* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 22.2 Features

- Debug enable/disable configuration
- Re-enable mechanism to enable debug
- Test mode enable/disable configuration
- Re-enable mechanism to enable test mode
- Mass erase enable/disable configuration
- SEC registers lock mechanism
- Boot configuration

## 22.3 Overview

The Security Controller (SEC) provides security control registers.

## 22.4 Functional Description

### 22.4.1 NVR Load

Part of SEC registers' values will be loaded from the Non-volatile Region (NVR). Refer to the NVR section for detailed mapping information.

### 22.4.2 Registers Write Lock

SEC registers could be locked by the RWL field to avoid unexpected write access. If the RWL feature is enabled by register write, then SEC registers remain locked until the next reset. If this lock function is implemented by NVR load, then SEC registers are always forbidden to write. If SEC registers are in the lock state, write access would result in a hard fault.

### 22.4.3 Debug Re-Enable Mechanism

Debug Re-Enable mechanism could be used when debug is disabled, and SEC registers are locked by NVR. If debug re-enable mechanism is allowed, the user could perform a 32-bit write to DEN register with the corresponding DEK0/1/2/3 value sequence. Only if four 32-bit writes match the values, debug will be enabled. In addition, this mechanism could also enable write access to SEC registers.

If DEA is not 0x5, this re-enable mechanism is disabled.

***Note:*** This feature is only valid in non-secure boot mode.

### 22.4.4 Test Mode Re-enable Mechanism

Test mode re-enable mechanism could be used when test mode is disabled, and SEC registers are locked by the NVR. When the test mode re-enable mechanism is allowed, the user could perform a 32-bit write to TME register with the corresponding TMEK value. Only if the 32-bit write matches the value, test mode will be enabled.

If TMEA is not 0x5, this re-enable mechanism is disabled.

### 22.4.5 Mass Erase Disable

Mass erase could be disabled in the secure state by setting MED to 1. If mass erase is disabled in the secure state and SEC registers are also locked by RWL, the method to re-enable mass erase is to use DEK to re-enable debug, then mass erase is also re-enabled.

***Note:***

- If Mass erase is disabled and Debug is disabled, the mass erase through SYS-AP is not possible. The application code needs to handle flash erase/update etc. control carefully; otherwise, the chip will not be able to update flash code anymore.
- This feature is only valid in non-secure boot mode.

### 22.4.6 Chip Unlock

The user could apply specific NVR configuration to disable debug, test mode, SEC reg write access and mass erase for security and safety requirements. However, sometimes there may be debugging requests to unlock these features. In this case, the user could use the debug re-enable mechanism, which could enable debug port and SEC registers feature.

***Note:*** This feature is only valid in non-secure boot mode.

### 22.4.7 Secure Boot

The FC7240F2M supports three boot modes, including Secure Boot, Non-secure Boot HSM Part, and Non-secure Boot Co-processor Part. The field secure_boot_en in NVR is used to select boot mode. If secure_boot_en is 0x0, secure boot mode is chosen. If not, non-secure boot mode is active.

***Note:*** The field secure_boot_en_backup should always keep the same value as secure_boot_en. Otherwise, the chip would fall into the lifecycle IN FIELD.

There are 5 lifecycles in secure boot mode. Each lifecycle has different access control of the chip. Refer to the FC7240 system security document for details.

Every lifecycle has its own slot in NVR to indicate its status, as the table below shows. Only bit[255:192] and bit[63:0] in slot are used for status judgment. The value of bit[191:64] has no impact.

**Table 55. Lifecycle status table**

| Lifecycle[255:192] | Lifecycle[63:0] | Status |
|---|---|---|
| Erased | Erased | Erased |
| Erased | Marked | Active |
| Marked | Marked | Inactive |
| Others | Others | Illegal |

Here marked indicates 0xFC20_11FE_C001_BEEF, erased indicates 0xFFFF_FFFF_FFFF_FFFF.

The chip lifecycle is decided by the combination of five lifecycle slots, as the table below shows.

**Table 56. Lifecycle table**

| Slot OEM DEV | Slot OED PDT | Slot IN FIELD | Slot SWFA | Slot HWFA | LC Status |
|---|---|---|---|---|---|
| Active | Erased | Erased | Erased | Erased | OEM DEV |
| Inactive | Active | Erased | Erased | Erased | OEM PDT |
| Inactive | Inactive | Active | Erased | Erased | IN FIELD |
| Inactive | Inactive | Inactive | Active | Erased | SWFA |
| Inactive | Inactive | Inactive | Inactive | Active | HWFA |
| Others | | | | | IN FIELD |

### 22.4.8 Non-Secure Boot HSM Part

The Non-secure Boot HSM Part mode is a new boot mode, the FlexCore is allocated to HSM engine in this mode. The field NVR_FLEXCORE_EN in NVR is used to select the HSM part. If NVR_FLEXCORE_EN is 0xF or 0x5, the Non-Secure Boot HSM Part is selected.

### 22.4.9 Non-Secure Boot Co-processor Part

The Non-secure Boot Co-processor Part mode is similar to the FC41xx series. If NVR field NVR_FLEXCORE_EN is not 0xF and 0x5, the Non-Secure Boot Co-processor Part is selected. The table below shows the difference in chip control between Secure Boot, Non-secure Boot HSM Part, and Non-secure Boot Co-processor Part.

**Table 57. Difference between Secure Boot, Non-secure Boot HSM Part and Non-secure Boot Co-processor Part**

| Access Type | Secure Boot | Non-Secure Boot HSM Part | Non-secure Boot Co-processor Part |
|---|---|---|---|
| Debug Port | Lifecycle/Debug Auth | FSECx/DEN/Mass Erase/DEK | FSECx/DEN/Mass Erase/DEK |
| Test Mode | TME | TME | TME |
| Mass Erase | Lifecycle | MED | MED |
| NVR Access | Lifecycle | NWP/NEP | NWP/NEP |
| NVR Key Access | lifecycle | NKRP/NWP/NEP | NKRP&(~NVR_DIS)/NWP/NEP |
| NVR0 Access | Lifecycle | ~ROM_DIS | ~ROM_DIS |
| NVR1 Access | Lifecycle | Access Prohibited | Access Prohibited |
| NVR2 Access | Lifecycle | UKAE | Access Prohibited |
| Boot Address | Rom Base | CPU0VTOR (SCM Register)/ Rom base | CPU0VTOR (SCM Register)/ Rom base |
| ISP Mode | ISP Pin/ISP Auth | ISP Pin & ISP Pin EN/ SOFT_EN_ISP_FLAG | ISP Pin & ISP Pin EN/ SOFT_EN_ISP_FLAG |

### 22.4.10 Chip Boot Address

In Non-secure Boot HSM Part and Non-secure Boot Co-processor Part mode, the user can configure BOOTROM bit to choose whether to boot from ROM.

- If BOOTROM is set to 1, the chip will boot from the ROM base and the boot code developed by Flagchip will be executed first. At the end of ROM code, the read access of the low 64K ROM will be disabled. Then code jumps to the user code address specified by *bootloader_address* in the NVR.
- If BOOTROM is set to 0, the boot address is decided by the register CPU0VTOR in the SCM. In this case, the low 64K ROM is disabled to be read by hardware. So, it is forbidden to configure CPU0VTOR to the low 64K ROM address when BOOTROM is 0.

In Secure Boot mode, the chip always boots from ROM.

## 22.5 Register Definition

SEC base address : 4001_4000h

**Register Overview - SEC**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | SEC_DEN (Debug Enable Register) | 32 | RW | 0000_000Ah | page |
| **4h** | SEC_FSEC0 (System Security_Control Register0) | 32 | RO | 0000_FFFFh | page |
| **8h** | SEC_FSEC1 (System Security_Control Register1) | 32 | RO | 0000_FFFFh | page |
| **Ch** | SEC_DCWOR (Debug Control Write Once Register) | 32 | RW | 0000_005Ah | page |

| | | | | | |
|---|---|---|---|---|---|
| **10h+n*4h** | SEC_DEKn (Debug Re-Enable Key Register) | 32 | WO | 0000_0000h | page |
| **20h** | SEC_TME (Test Mode Enable Register) | 32 | RW | 0000_0005h | page |
| **24h** | SEC_TMEA (Test Mode Re-Enable Allow Register) | 32 | RW | 0000_0005h | page |
| **28h** | SEC_TMEK (Test Mode Re-Enable Key Register) | 32 | WO | FFFF_FFFFh | page |
| **30h** | SEC_FCR0 (Flash Control Register0) | 32 | RW | 0000_FF0Fh | page |
| **3Ch** | SEC_NKRP (NVR Key Read Protection) | 32 | RW | 0000_000Fh | page |
| **60h** | SEC_BCS (Boot Configuration Status Register) | 32 | RO | 6000_018Fh | page |
| **64h** | SEC_UKAC (User Key Access Configuration Register) | 32 | RO | 0000_000Fh | page |
| **68h** | SEC_BRC0 (BootROM Configuration Register0) | 32 | RW | 0000_0FFFh | page |
| **6Ch** | SEC_BRC1 (BootROM Configuration Register1) | 32 | RO | 01FF_FFF0h | page |
| **70h** | SEC_SW_CFG_ISP_FLG (Software Configuration to Enter ISP Flag Register) | 32 | RW | 0000_0000h | page |
| **74h** | SEC_BRC2 (BootROM Configuration Register2) | 32 | RO | FE00_000Fh | page |
| **78h** | SEC_IMGEA (Image Address Register) | 32 | RO | FFFF_FFFCh | page |
| **7Ch** | SEC_NVR_VER (NVR Version Register) | 32 | RW | 00FF_FFFFh | page |
| **88h** | SEC_LCSTAT (Lifecycle Status Register) | 32 | RO | 0000_0000h | page |
| **8Ch** | SEC_FAC (Flash Access Control Register) | 32 | RW | 0000_0000h | page |
| **94h** | SEC_FLEXCORE_EN (FlexCore Enable Register) | 32 | RW | 0000_000Fh | page |
| **98h** | SEC_FLEX_CODE_ADDR (Code Address for FlexCore Register) | 32 | RW | 3FFF_FFFCh | page |
| **9Ch** | SEC_PFLASH_PRLLL_EN (PFLASH Parallel Enable Register) | 32 | RW | 0000_000Fh | page |

### 22.5.1 SEC_DEN (Debug Enable Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | DEN | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

**SEC_DEN field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**DEN** | RW | The SSC0/SSC1 state will be loaded to DEN register. If system is unsecure, DEN bits will be set to 0x5 (debug enable), otherwise, DEN bits will be set to 0xA. If mass erase is done successfully, DEN should be set to 0x5. If entering test mode, it will be forced to enable debug.<br>After reset, DEN will be reset to 0xA (debug disable). DEN will use to gate the ARM AHBAP function. If RWL is 0x5, DEN can be written at will. If RWL is not 0x5, it depends on DEA setting. If DEA is 0x5, it can use re-enable key to re-enable the debug. |

### 22.5.2 SEC_FSEC0 (System Security_Control Register0)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SSC0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**SEC_FSEC0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**SSC0** | RO | The value will be loaded from NVR. Refer to the NVR section for detailed mapping information. Only SSC0 = 0xC35A & SSC1 = 0xFFFF means system is unsecured, all other values mean system is secured. After reset, both registers will be reset to 0xFFFF. |

### 22.5.3 SEC_FSEC1 (System Security_Control Register1)

Offset : 8h

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

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SSC1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**SEC_FSEC1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**SSC1** | RO | The value will be loaded from NVR. Refer to the NVR section for detailed mapping information. Only SSC0 = 0xC35A & SSC1 = 0xFFFF means system is unsecured, all other values mean system is secured. After reset, both registers will be reset to 0xFFFF. |

### 22.5.4 SEC_DCWOR (Debug Control Write Once Register)

Initial value will be loaded from NVR sector. Refer to the NVR section for detailed mapping information.

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RWL | | | | DEA | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |

**SEC_DCWOR field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**DEA** | RW | Debug Re-Enable Allow<br>0x5 allows re-enable function; others mean disable. |

| | | |
|---|---|---|
| **7-4**<br>**RWL** | RW | Register Write Lock<br>0x5 or 0xF allows register write.<br>If RWL is in lock state, DEA cannot be changed by software. RWL is also used to lock the other control register. |

### 22.5.5 SEC_DEKn (Debug Re-Enable Key Register)

Four keys loaded from NVR sector, Refer to the NVR section for detailed mapping information.

Offset : 10h+N*4h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | DEK | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | DEK | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SEC_DEKn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**DEK** | WO | Debug Re-Enable Key<br>To re-enable the DEN while RWL is locked, if DEA is not 0x5, DEN is not allowed to be changed by software.<br>If DEA is 0x5, user should perform a 32-bit write to DEN register with the corresponding DEK0/1/2/3 value sequence. Only if four 32-bit writes match, it will set the DEN to 0x5 to enable debug and unlock SEC registers. If the check fails at the last step write, it will block the debug write capability until the next reset.<br>DEK0/1/2/3 are write-only registers. If RWL is in unlock state, software can change the setting. |

### 22.5.6 SEC_TME (Test Mode Enable Register)

Value loaded from NVR sector. Refer to the NVR section for detailed mapping information.

Offset : 20h

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

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | TME | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

**SEC_TME field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**TME** | RW | Test Mode Enable<br>Test Mode access allow.<br>0xF or 0x5 means enable; others mean disable. |

### 22.5.7 SEC_TMEA (Test Mode Re-Enable Allow Register)

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | TMEA | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

**SEC_TMEA field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**TMEA** | RW | Test Mode Re-enable Allow (use TestMode re-enable key to enable test mode, 0xF or 0x5 allow TestMode write).<br>After reset TME/TMEA will be reset to 0x5, TME will use to gate the test-related control and test mode entry. If RWL is in unlock state, software can change the register at will. If RWL is in lock state, depending on the TME setting, if TMEA is 0xF or 0x5, writing the correct key to TME will enable the TestMode entry. |

### 22.5.8 SEC_TMEK (Test Mode Re-Enable Key Register)

Offset : 28h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | TMEK | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | TMEK | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**SEC_TMEK field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**TMEK** | WO | Test Mode Re-enable Key<br>Write only.<br>If RWL is in unlock state, software can change the register. |

### 22.5.9 SEC_FCR0 (Flash Control Register0)

Value loaded from NVR sector. Refer to the NVR section for detailed mapping information. If RWL is in unlock state, software can change the register.

Offset : 30h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NEP | | | | NWP | | | | 0 | | | | MED | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**SEC_FCR0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**MED** | RW | Mass Erase Disable<br>0xF or 0x5 means mass erase enable; others mean disable. |
| **11-8**<br>**NWP** | RW | Block 0 NVR Write Protection<br>0x5 or 0xF means no program protection. |
| **15-12**<br>**NEP** | RW | Block 0 NVR Erase Protection<br>0x5 or 0xF means no erase protection. |

### 22.5.10 SEC_NKRP (NVR Key Read Protection)

Offset : 3Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | NKRP | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**SEC_NKRP field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**NKRP** | RW | NVR Key Read Protection<br>0x5 or 0xF means no read protection.<br>If key read protection is valid, the protection can be removed by writing the right Block Cipher Key 0~7 to this register while the RWL is in unlock state. |

### 22.5.11 SEC_BCS (Boot Configuration Status Register)

Offset : 60h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | PF_128BIT_MODE | PART_MODE | 0 | | ISPMODE | BOOTROM | 0 | NMIDIS | FBS |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**SEC_BCS field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**FBS** | RO | Fast Boot Select<br>0b - Select PLL0 as the core clock source; the core clock is 240 MHz.<br>1b - Select FIRC as the core clock source; the core clock is 96 MHz. |
| **1**<br>**NMIDIS** | RO | NMI Function Enable/Disable<br>0b - NMI pin function is enabled. The associated pin defaults to internal pull up enabled.<br>1b - NMI pin function is disabled after reset. |
| **3**<br>**BOOTROM** | RO | Boot ROM Configuration<br>0b - Boot from GPR defined address (except ROM).<br>1b - Boot from ROM. |
| **4**<br>**ISPMODE** | RO | ISP Mode Status<br>0b - ISP mode is inactive.<br>1b - ISP mode is active. |
| **7**<br>**PART_MODE** | RO | Chip Part Mode<br>0b - Chip is in FlexCore part mode.<br>1b - Chip is in HSM part mode. |
| **8**<br>**PF_128BIT_MODE** | RO | PFlash 128-bit Mode Active Flag<br>0b - PFlash is in non-parallel mode (64-bit).<br>1b - PFlash is in parallel mode (128-bit). |

### 22.5.12 SEC_UKAC (User Key Access Configuration Register)

Value loaded from NVR sector. Refer to the NVR section for detailed mapping information.

Offset : 64h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | UKAE | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**SEC_UKAC field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**UKAE** | RO | User Key Access Enable<br>Only valid in non-secure boot.<br>0x5/0xF - User key can be read/programmed/erased by host CPU.<br>Others - User key is not available for host CPU. |

### 22.5.13 SEC_BRC0 (BootROM Configuration Register0)

Offset : 68h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | BOOT_SUCCESS | 0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | ISP_AUTH_EN | | | | DEBUG_AUTH_EN | | | | SECURE_BOOT_DIS | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**SEC_BRC0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**SECURE_BOOT_DIS** | RO | Secure Boot Disable<br>Value loaded from NVR sector.<br>0x0b - Secure boot mode.<br>Others - Non secure boot mode. |

| | | |
|---|---|---|
| **7-4**<br>**DEBUG_AUTH_EN** | RO | Host Debug Auth Enable<br>Only valid in secure boot. (Value loaded from NVR sector)<br>0x5/0xF - Host debug authentication enable.<br>Others - Host debug authentication disable. |
| **11-8**<br>**ISP_AUTH_EN** | RO | ISP Auth Enable<br>Only valid in secure boot. (Value loaded from NVR sector)<br>0x5/0xF - ISP authentication enable.<br>Others - ISP authentication disable. |
| **18**<br>**BOOT_SUCCESS** | RW | User Code Boot Success<br>Writing this bit will enable the FlexCore user code. Only valid in FlexCore mode. |

### 22.5.14 SEC_BRC1 (BootROM Configuration Register1)

Value loaded from NVR sector. Refer to the NVR section for detailed mapping information.

Offset : 6Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | SW_CFG_ISP_EN | ROM_LOOP_CTRL | KEY_PROT | ISPAIRM | ISP_PIN_EN | DBKEA | | | NON_SEC_VERIF_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DBK_IN_EN | | | | OSCA | OSCFREQ | | | CANBR | | UARTBR | | 0 | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

**SEC_BRC1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**UARTBR** | RO | FCUART Baud Rate for ISP<br>00b - 2Mbps<br>01b - 1Mbps<br>10b - 512Kbps<br>11b - 115200bps |

| | | |
|---|---|---|
| **7-6**<br>**CANBR** | RO | CAN Baud Rate for ISP<br>00b - CANFD 4Mbps/CAN 1Mbps if CANFD is not valid.<br>01b - CANFD 2Mbps/CAN 1Mbps if CANFD is not valid.<br>10b - CAN 1Mbps<br>11b - CAN 500Kbps |
| **10-8**<br>**OSCFREQ** | RO | OSC Frequency<br>000b - 8MHz<br>001b - 12MHz<br>010b - 16MHz<br>011b - 24MHz<br>100b - 25MHz<br>101b - 32MHz<br>110b - 48MHz<br>111b - 40MHz |
| **11**<br>**OSCA** | RO | OSC Available<br>0b - OSC is available<br>1b - OSC is not available |
| **15-12**<br>**DBK_IN_EN** | RO | Debug Backdoor Key Input Enable<br>0011b - User can input DBK through debug mailbox, ISP is not valid.<br>1100b - User can input DBK through ISP, debug mailbox is not valid.<br>1111b - User can input DBK through both debug mailbox and ISP.<br>Other Values - User cannot input DBK through both debug mailbox and ISP. |
| **16**<br>**NON_SEC_VERIF_EN** | RO | User Code Verification Enable under Non-secure Boot Mode<br>0b - User code verification enable in non-secure boot mode<br>1b - User code verification disable in non-secure boot mode |
| **19-17**<br>**DBKEA** | RO | Debug Backdoor Key Encryption Algorithm<br>000b - AES256-CBC<br>111b - SM4-CBC |
| **20**<br>**ISP_PIN_EN** | RO | ISP Pin Disable/Enable<br>Note: Only valid in non-secure mode.<br>0b - ISP pin is enabled during pin reset<br>1b - ISP pin is disabled |
| **21**<br>**ISPAIRM** | RO | ISP Mode Allow in Recovery Mode<br>0b - ISP mode is not allowed in recovery mode<br>1b - ISP mode is allowed in recovery mode |
| **22**<br>**KEY_PROT** | RO | Enable the Output of Plaintext Decrypted ROOT Key<br>0b - The output of plaintext decrypted ROOT key is enabled<br>1b - The output of plaintext decrypted ROOT key is disabled |

| | | |
|---|---|---|
| **23**<br>**ROM_LOOP_CTRL** | RO | ROM Loop Control Flag<br>0b - ROM executes normal run under VIRGIN and FT state.<br>1b - ROM enters debug loop under VIRGIN and FT state. |
| **24**<br>**SW_CFG_ISP_EN** | RO | Enable Software Configuration to Enter ISP<br>0b - Software configuration to enter ISP is enabled.<br>1b - Software configuration to enter ISP is disabled. |

### 22.5.15 SEC_SW_CFG_ISP_FLG (Software Configuration to Enter ISP Flag Register)

Offset : 70h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | SW_CFG_ISP_FLG |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SEC_SW_CFG_ISP_FLG field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**SW_CFG_ISP_FLG** | RW | Software Configuration to Enter ISP Flag<br>Writing 0xC001_BEEF to this register will set this bit. Writing other values will clear this bit. After this bit is set and SOFT_EN_ISP is 1'b0, trigger a non-POR reset, then it will enter ISP mode.<br>0b - Software configuration to enter ISP flag is not set.<br>1b - Software configuration to enter ISP flag is set. |

### 22.5.16 SEC_BRC2 (BootROM Configuration Register2)

Value loaded from NVR sector, Refer to the NVR section for detailed mapping information.

Offset : 74h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DECRP_ALG | USERCODE_VERIFICATION_ALG | | | ISP_INST_SEL | | | USERCODE_VERIFY_MASK | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | USERCODE_VERIFY_MASK | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**SEC_BRC2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **24-0**<br>**USERCODE_VERIFY_MASK** | RO | User Code Verification Mask<br>This mask contains some information for boot ROM, as below shows.<br>{C1_C2_reset, RGM_SRS[31:29], RGM_C0_SRS[20:0]} |
| **27-25**<br>**ISP_INST_SEL** | RO | ISP Instance Selection<br>000b - FCUART1<br>001b - FCUART2<br>010b - FLEXCAN2<br>011b - FLEXCAN3<br>111b - No Select |
| **30-28**<br>**USERCODE_VERIFICATION_ALG** | RO | User Code Verification Algorithm<br>000b - AES256-CMAC (only available when FlexCore enable)<br>001b - ECC256 + SHA256 (only available when FlexCore disable)<br>010b - SM4-CMAC (only available when FlexCore enable)<br>011b - RSA4096 + SHA256 (only available when FlexCore enable)<br>111b - SM2 + SM3 (only available when FlexCore disable) |
| **31**<br>**DECRP_ALG** | RO | Debug/ISP/SWFA Authentication and USRK Decryption Algorithm<br>0b - ECC256<br>1b - SM2 |

### 22.5.17 SEC_IMGEA (Image Address Register)

Value loaded from NVR sector. Refer to the NVR section for detailed mapping information.

Offset : 78h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | IMAGE_ADDR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | IMAGE_ADDR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

**SEC_IMGEA field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0<br>IMAGE_ADDR** | RO | Image Address |

### 22.5.18 SEC_NVR_VER (NVR Version Register)

Value loaded from NVR sector. Refer to the NVR section for detailed mapping information.

Offset : 7Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CHIP_VER | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | V_VER | | | | R_VER | | | | | | C_VER | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**SEC_NVR_VER field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-0<br>C_VER** | RW | C Version |
| **11-6<br>R_VER** | RW | R Version |
| **15-12<br>V_VER** | RW | V Version |

| | | |
|---|---|---|
| **23-16**<br>**CHIP_VER** | RW | Chip Version |

### 22.5.19 SEC_LCSTAT (Lifecycle Status Register)

Value loaded from NVR sector. Refer to the NVR section for detailed mapping information.

Offset : 88h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | LIFECYCLE_HWFA | LIFECYCLE_SWFA | LIFECYCLE_IN_FIELD | LIFECYCLE_OEM_PDT | LIFECYCLE_OEM_DEV |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SEC_LCSTAT field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**LIFECYCLE_OEM_DEV** | RO | Lifecycle OEM Development<br>0b - Chip is not in this lifecycle<br>1b - Chip is in this lifecycle |
| **1**<br>**LIFECYCLE_OEM_PDT** | RO | Lifecycle OEM Production<br>0b - Chip is not in this lifecycle<br>1b - Chip is in this lifecycle |
| **2**<br>**LIFECYCLE_IN_FIELD** | RO | Lifecycle In Field<br>0b - Chip is not in this lifecycle<br>1b - Chip is in this lifecycle |
| **3**<br>**LIFECYCLE_SWFA** | RO | Lifecycle Software Fault Analysis<br>0b - Chip is not in this lifecycle<br>1b - Chip is in this lifecycle |

| | | |
|---|---|---|
| **4**<br>**LIFECYCLE_HWFA** | RO | Lifecycle Hardware Fault Analysis<br>0b - Chip is not in this lifecycle<br>1b - Chip is in this lifecycle |

### 22.5.20 SEC_FAC (Flash Access Control Register)

Offset : 8Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | HSM_NVR_EPROT | HSM_NVR_WPROT | HSM_NVR_RPROT | HSM_UKEY_EPROT | HSM_UKEY_WPROT | HSM_UKEY_RPROT | 0 | | HOST_NVR_EPROT | HOST_NVR_WPROT | HOST_NVR_RPROT | HOST_UKEY_EPROT | HOST_UKEY_WPROT | HOST_UKEY_RPROT |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SEC_FAC field descriptions**

| Field | Type | Description |
|---|---|---|
| **16**<br>**HOST_UKEY_RPROT** | RO | Host User Key Read Protection<br>0b - Host read access to User Key region is enabled<br>1b - Host read access to User Key region is disabled |
| **17**<br>**HOST_UKEY_WPROT** | RO | Host User Key Write Protection<br>0b - Host program access to User Key region is enabled<br>1b - Host program access to User Key region is disabled |
| **18**<br>**HOST_UKEY_EPROT** | RO | Host User Key Erase Protection<br>0b - Host erase access to User Key region is enabled<br>1b - Host erase access to User Key region is disabled |
| **19**<br>**HOST_NVR_RPROT** | RO | Host NVR Key Read Protection<br>0b - Host read access to NVR key region is enabled<br>1b - Host read access to NVR key region is disabled |
| **20**<br>**HOST_NVR_WPROT** | RO | Host NVR Write Protection<br>0b - Host program access to NVR region is enabled<br>1b - Host program access to NVR region is disabled |

| | | |
|---|---|---|
| **21**<br>**HOST_NVR_EPROT** | RO | Host NVR Erase Protection<br>0b - Host erase access to NVR region is enabled<br>1b - Host erase access to NVR region is disabled |
| **24**<br>**HSM_UKEY_RPROT** | RO | HSM User Key Read Protection<br>0b - HSM Read access to User Key region is enabled<br>1b - HSM Read access to User Key region is disabled |
| **25**<br>**HSM_UKEY_WPROT** | RO | HSM User Key Program Protection<br>0b - HSM Program access to User Key region is enabled<br>1b - HSM Program access to User Key region is disabled |
| **26**<br>**HSM_UKEY_EPROT** | RO | HSM User Key Erase Protection<br>0b - HSM erase access to User Key region is enabled<br>1b - HSM erase access to User Key region is disabled |
| **27**<br>**HSM_NVR_RPROT** | RO | HSM NVR Key Read Protection<br>0b - HSM Read access to NVR key region is enabled<br>1b - HSM Read access to NVR key region is disabled |
| **28**<br>**HSM_NVR_WPROT** | RO | HSM NVR Program Protection<br>0b - HSM Program access to NVR region is enabled<br>1b - HSM Program access to NVR region is disabled |
| **29**<br>**HSM_NVR_EPROT** | RO | HSM NVR Erase Protection<br>0b - HSM Erase access to NVR region is enabled<br>1b - HSM Erase access to NVR region is disabled |

### 22.5.21 SEC_FLEXCORE_EN (FlexCore Enable Register)

Offset : 94h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | FLEX CORE _DBG _EN | FLEXCORE_EN | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**SEC_FLEXCORE_EN field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**FLEXCORE_EN** | RW | FlexCore Enable<br>0xF or 0x5 means FlexCore disable and HSM core enable; other values mean FlexCore enable and HSM core disable.<br>While NVR is invalid, this bit field 3:0 can be written as will. When NVR is valid, the value of this field is loaded from NVR, writing to this field takes no effect. After writing this register to enable FlexCore mode, the FlexCore subsystem should be reset manually by setting and clearing the SCM_SUBSYS_PCC[SWRST]. |
| **4**<br>**FLEXCORE_DBG_EN** | RW | FlexCore Debug Enable<br>This field takes effect when FlexCore is enabled.<br>0b - Disable the FlexCore debug.<br>1b - Enable the FlexCore debug. |

### 22.5.22 SEC_FLEX_CODE_ADDR (Code Address for FlexCore Register)

Offset : 98h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | FLEX_CODE_ADDR | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FLEX_CODE_ADDR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

**SEC_FLEX_CODE_ADDR field descriptions**

| Field | Type | Description |
|---|---|---|
| **29-0**<br>**FLEX_CODE_ADDR** | RW | Code Header Address for FlexCore<br>Always 4-byte aligned. |

### 22.5.23 SEC_PFLASH_PRLLL_EN (PFLASH Parallel Enable Register)

Offset : 9Ch

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

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | PFLASH_PRLLL_EN | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**SEC_PFLASH_PRLLL_EN field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**PFLASH_PRLLL_EN** | RW | PFLASH Parallel Enable<br>1111b - Enable<br>0101b - Disable<br>Others - Enable |

Flagchip

# Chapter 23 CORDIC

## 23.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

This chapter describes the CORDIC module in the following sections:

- Chip-specific Information
- Features
- Overview
- Functional Description
- Registers

## 23.2 Chip-specific Information

N/A

## 23.3 Features

The CORDIC module has the below features:

- Configurable calculation iteration of 8, 16, and 24. The more iterations, the higher the accuracy. When you configure CORDIC in 24 iteration mode, the calculation accuracy can reach 7 decimal point places.
- Readable status register indicates if the CORDIC is busy or has finished the operation.
- Configurable interrupt. An interrupt is generated after one calculation is completed if the interrupt is enabled.
- Supports scale feature when number exceeds the fixed point range. Some of the functions have a domain which exceeds the fixed point numerical range. In this case a power of two scaling factor (right shift) can be applied. The scaling factor must be accounted for in the CORDIC input and undone at the output.
- CORDIC has two modes of rotation and vector, which can be used in the circular coordinate system, linear coordinate system, and hyperbolic coordinate system respectively. Through this principle, we can realize many complex operations, they are:

  a) Trigonometric function
  b) Multiplication function
  c) Division function
  d) Hyperbolic function
  e) Logarithmic function
  f) Exponential function

  And some other operations can be realized via the combination of the operations above.

- Three input registers and three output registers to save the data.

## 23.4 Overview

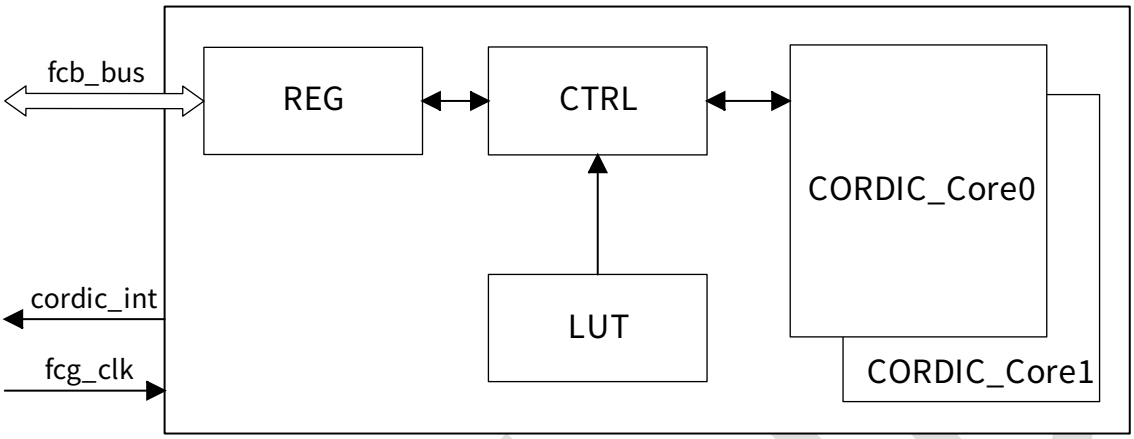
CORDIC is a simplified algorithm module, which transforms many complex operations into an iterative operation of only shift and addition. It can accelerate some operations several hundred times.

Flagchip

## 23.5 Block Diagram

The following is the block diagram of the CORDIC module.

**Figure 36. CORDIC block diagram**




## 23.6 Functional Description

### 23.6.1 Numerical Representation

The CORDIC module operates in fixed point representation as shown below.

| 31 | 30:0 |
|---|---|
| **Sign bit** | Fractional bit |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | $A_{30}$ | $A_{29}$ | $A_{28}$ | $A_{27}$ | $A_{26}$ | $A_{25}$ | $A_{24}$ | $A_{23}$ | $A_{22}$ | $A_{21}$ | $A_{20}$ | $A_{19}$ | $A_{18}$ | $A_{17}$ | $A_{16}$ | $A_{15}$ | $A_{14}$ | $A_{13}$ | $A_{12}$ | $A_{11}$ | $A_{10}$ | $A_9$ | $A_8$ | $A_7$ | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ |

$$V_{float} = (-1)^s \sum_{n=0}^{30} \frac{1}{2^{31-n}} A_n$$

The numeric range is from -1(0x80000000) to $1-2^{-31}$(0x7FFFFFFF), and the precision is $2^{-31}$.

For angle representation, angles are expressed in radian and divided by $\pi$. Hence -1 corresponds to the angle $-\pi$ and +1 corresponds to $+\pi$

$$\theta_{radian} = \frac{\theta}{180}$$

The software needs to convert a floating-point number to a fixed-point number when using the CORDIC to operate a function. It is suggested to use FPU to convert the floating-point number to a fixed-point number, if the device supports FPU. If not, please covert the floating-point number to a fixed-point number by software.

### 23.6.2 CORDIC Function Summary

Steps to initiate an operation:

1) Write the data input registers with values.
2) Configure the CORDIC module to a specific mode.
3) Wait for the STAT[DONE] flag asserted.
4) Read the result from the data output registers.

#### 23.6.2.1 cos and sin

If programming X_INPUT with $\frac{1}{K}$, Y_INPUT with 0 and Z_INPUT with an angle $\theta$(divided by $\pi$), then users can get the result of $\cos(z)$ and $\sin(z)$ from X_OUTPUT and Y_OUTPUT. If any of the value is equal to or greater than 1, a scale must be applied in software to adapt it to (-1,1).

| Parameter | Description | Range |
|---|---|---|
| $\theta$ | angle | $(-90°, +90°)$ |
| X_I | $\frac{1}{K}(K = 1.6467602581)$ | 0.6072529350 |
| Y_I | 0 | 0 |
| Z_I | Angle $\theta$ in radians, divided by $\pi$ | $(-\frac{1}{2}, \frac{1}{2})$ |
| X_O | $\cos(z)$ | $(-1,1)$ |
| Y_O | $\sin(z)$ | $(-1,1)$ |

#### 23.6.2.2 Arctan and modulus

To calculate angle or modulus of a vector [x y], users can program X_INPUT with x coordinate, Y_INPUT with y coordinate and Z_INPUT with 0. Then users can get $\arctan(y/x)$ from Z_OUTPUT and get $K\sqrt{x^2+y^2}$ from X_OUTPUT.

If any of the value is equal to or greater than 1, a scale must be applied in software to adapt it to (-1,1).

| Parameter | Description | Range |
|---|---|---|
| X_I | x coordinate | $(0, \frac{1}{K})$ |
| Y_I | y coordinate | $(-\frac{1}{K}, \frac{1}{K})$ |
| Z_I | 0 | 0 |
| X_O | $K\sqrt{x^2+y^2}$ | $(0,1)$ |
| Z_O | $\arctan(y/x)$, phase angle $\theta$ in radians divided by $\pi$ | $(-\frac{1}{2}, \frac{1}{2})$ |

#### 23.6.2.3 Multiply

To calculate $a * b$, users can program X_INPUT with $a$, Y_INPUT with 0 and Z_INPUT with $b$. Then users can get the result of $a * b$ from Y_OUTPUT.

| Register | Description | Range |
|---|---|---|
| X_I | $a$ | $(-1,1)$ |
| Y_I | 0 | 0 |
| Z_I | $b$ | $(-1,1)$ |
| Y_O | $a * b$, multiply result | $(-1,1)$ |

#### 23.6.2.4 Division

To calculate of $b/a$, users can program X_INPUT with $a$, Y_INPUT with $b$, and Z_INPUT with 0. Then users can get the result of $b/a$ from Z_OUTPUT. If any of the value is equal to or greater than 1, a scale must be applied in software to adapt it to (-1,1).

| Parameter | Description | Range |
|---|---|---|
| X_I | $a$ | $\|b\| \leq \|a\|$ |
| Y_I | $b$ | $\|b\| \leq \|a\|$ |
| Z_I | 0 | 0 |
| Z_O | $b/a$, division result | $(-1,1)$ |

#### *23.6.2.5 cosh and sinh*

To calculate $cosh(z)$ and $sinh(z)$, users can program X_INPUT with $\frac{1}{K_a} * 2^{-n}$, Y_INPUT with 0 and Z_INPUT with $z * 2^{-n}$. Then users can get the result of $2^{-n} * cosh(z)$ and $2^{-n} * sinh(z)$ from X_OUTPUT and Y_OUTPUT. If any of the value is equal to or greater than 1, a scale must be applied in software to adapt it to (-1,1).

| Parameter | Description | Range |
|---|---|---|
| X_I | $\frac{1}{K_a} * 2^{-n} (K_a = 0.828159361)$ | 0.6037485339 |
| Y_I | 0 | 0 |
| Z_I | $z * 2^{-n}$ | $(-0.5590864481, 0.5590864481)$ |
| X_O | $2^{-n} * cosh(z)$ | $[0.5, 0.8465339990)$ |
| Y_O | $2^{-n} * sinh(z)$ | $(-0.6830957557, 0.6830957557)$ |
| SCALE | $n$ | 1 |

#### *23.6.2.6 arctanh*

To calculate $arctanh(a)$, users can program X_INPUT with $2^{-n}$, Y_INPUT with $a * 2^{-n}$, and Z_INPUT with 0. Then users can get the result of $2^{-n} * arctanh(a)$ from Z_OUTPUT. If any of the value is equal to or greater than 1, a scale must be applied in software to adapt it to (-1,1).

| Parameter | Description | Range |
|---|---|---|
| X_I | $1 * 2^{-n}$ | 0.5 |
| Y_I | $a * 2^{-n}$ | $(-0.4034662261, 0.4034662261)$ |
| Z_I | 0 | 0 |
| Z_O | $2^{-n} * arctanh(a)$ | $(-0.5590864481, 0.5590864481)$ |
| SCALE | $n$ | 1 |

#### 23.6.2.7 sqrt(w)

To calculate $sqrt(w)$, users can program X_INPUT with $(w+\frac{1}{4})*2^{-n}$, Y_INPUT with $(w-\frac{1}{4})*2^{-n}$, and Z_INPUT with 0. Then users can get the result of $2^{-n}K_a*\sqrt{w}$ from Z_OUTPUT. If any of the value is equal to or greater than 1, a scale must be applied in software to adapt it to (-1,1).

| Parameter | Description | Range |
|---|---|---|
| $w$ | $w$ | $(0.0267120594, 2.3397671865)$ |
| X_I | $(w+\frac{1}{4})*2^{-n}$ | $(0.0691780148, 0.6474417966)$ |
| Y_I | $(w-\frac{1}{4})*2^{-n}$ | $(-0.0558219851, 0.5224417966)$ |
| Z_I | 0 | 0 |
| X_O | $2^{-n}K_a*\sqrt{w}$ | $(0.0338382278, 0.3166943000)$ |
| SCALE | $n$ | 2 |

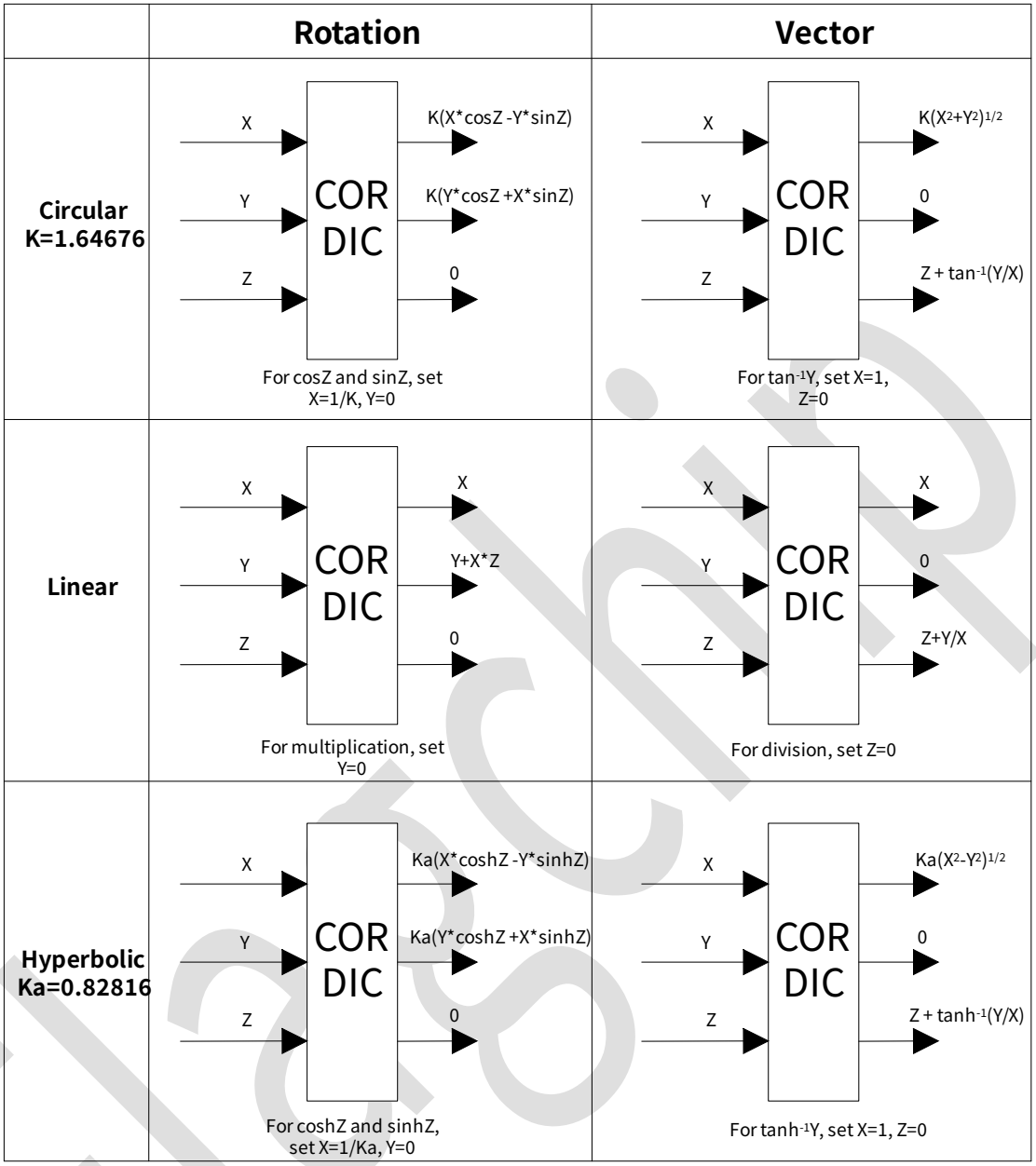
#### 23.6.2.8 ln(w)

To calculate $ln(w)$, users can program X_INPUT with $(w+1)*2^{-n}$, Y_INPUT with $(w-1)*2^{-n}$, and Z_INPUT with 0. Then users can get the result of $2^{-n-1}*ln(w)$ from Z_OUTPUT. If any of the value is equal to or greater than 1, a scale must be applied in software to adapt it to (-1,1).

| Parameter | Description | Range |
|---|---|---|
| $w$ | $w$ | $(0.1068482375, 9.3590687463)$ |
| X_I | $(w+1)*2^{-n}$ | $(0.0691780149, 0.6474417966)$ |
| Y_I | $(w-1)*2^{-n}$ | $(-0.0558219851, 0.5224417966)$ |
| Z_I | 0 | 0 |
| Z_O | $2^{-n-1}*ln(w)$ | $(-0.0698858060, 0.0698858060)$ |
| SCALE | $n$ | 4 |

The table below summarizes the basic CORDIC functions. It helps you with the application of the CORDIC.

**Figure 37. CORDIC application**




### 23.6.3 CORDIC Extended Functions

Although the CORDIC algorithm can only calculate a small number of functions directly, more functions can be calculated indirectly.

| | |
|---|---|
| $tanz = \frac{sinz}{cosz}$ | $cos^{-1}w = tan^{-1}\frac{\sqrt{1-w^2}}{w}$ |
| $tanhz = \frac{sinhz}{coshz}$ | $sin^{-1}w = tan^{-1}\frac{w}{\sqrt{1-w^2}}$ |
| $lnw = 2tanh^{-1}\frac{w-1}{w+1}$ | $cosh^{-1}w = ln\left(w+\sqrt{1-w^2}\right)$ |
| $e^z = sinhz + coshz$ | $sinh^{-1}w = ln\left(w+\sqrt{1+w^2}\right)$ |
| $w^t = e^{tlnw}$ | $\sqrt{w} = \sqrt{\left(w+\frac{1}{4}\right)^2 - \left(w-\frac{1}{4}\right)^2}$ |

## 23.7 Register Definition

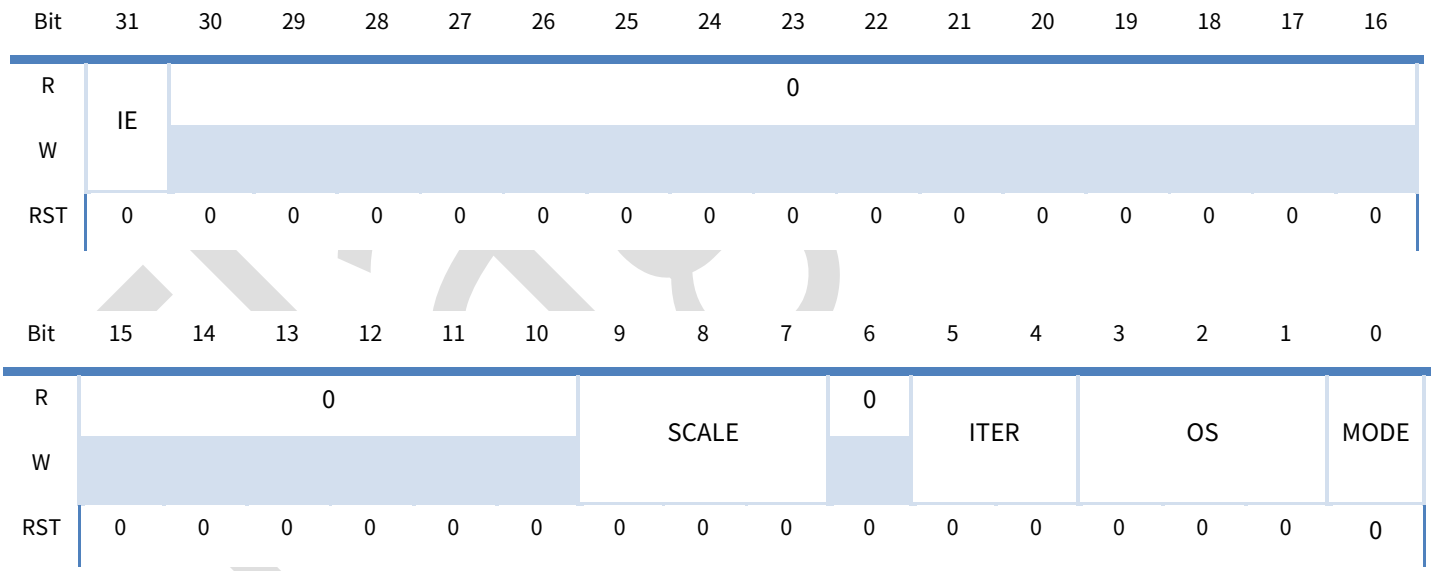
CORDIC base address : 4002_B000h

**Register Overview - CORDIC**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | CORDIC_CTRL (Control Register) | 32 | RW | 0000_0000h | page |
| **10h** | CORDIC_X_INPUT (X-axis Input Data Register) | 32 | RW | 0000_0000h | page |
| **14h** | CORDIC_Y_INPUT (Y-axis Input Data Register) | 32 | RW | 0000_0000h | page |
| **18h** | CORDIC_Z_INPUT (Z-axis Input Data Register) | 32 | RW | 0000_0000h | page |
| **20h** | CORDIC_STAT (State Register) | 32 | W1C | 0000_0000h | page |
| **24h** | CORDIC_X_OUTPUT (X-axis Output Data Register) | 32 | RO | 0000_0000h | page |
| **28h** | CORDIC_Y_OUTPUT (Y-axis Output Data Register) | 32 | RO | 0000_0000h | page |
| **2Ch** | CORDIC_Z_OUTPUT (Z-axis Output Data Register) | 32 | RO | 0000_0000h | page |

### 23.7.1 CORDIC_CTRL (Control Register)

Offset : 0h




**CORDIC_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**IE** | RW | Interrupt Enable<br>0b - Disable interrupt<br>1b - Enable interrupt |

| | | |
|---|---|---|
| **9-7**<br>**SCALE** | RW | Scale Factor<br>Some of the functions have a domain which exceeds the fixed point numerical range. In this case a power of two scaling factor (right shift) can be applied. The scaling factor must be accounted for in the CORDIC input and undone at the output.<br>000b - No scale<br>001b - Scale by 2<br>010b - Scale by 4<br>011b - Scale by 8<br>...<br>111b - Scale by 128 |
| **5-4**<br>**ITER** | RW | Iteration Number<br>00b - 8<br>01b - 16<br>10b - 24<br>11b - 24 |
| **3-1**<br>**OS** | RW | Operation System<br>000b - Trigonometric system<br>001b - Hyperbolic system<br>010b - Linear system<br>011b - Reserved<br>100b - Reserved<br>101b - Reserved<br>110b - Reserved<br>111b - Reserved |
| **0**<br>**MODE** | RW | Operation Mode<br>0b - Rotate mode<br>1b - Vector mode |

### 23.7.2 CORDIC_X_INPUT (X-axis Input Data Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CORDIC_X_INPUT field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 DATA** | RW | The input data for X-axis |

### 23.7.3 CORDIC_Y_INPUT (Y-axis Input Data Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CORDIC_Y_INPUT field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 DATA** | RW | The input data for Y-axis |

### 23.7.4 CORDIC_Z_INPUT (Z-axis Input Data Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CORDIC_Z_INPUT field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**DATA** | RW | The input data for Z-axis |

### 23.7.5 CORDIC_STAT (State Register)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | DONE |
| W | | | | | | | | | | | | | | | | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CORDIC_STAT field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**DONE** | W1C | Operation Done<br>The value of DONE indicates if the operation has been done. This flag needs to be cleared by software.<br>0b - Operation not started or not done<br>1b - Operation done |

### 23.7.6 CORDIC_X_OUTPUT (X-axis Output Data Register)

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CORDIC_X_OUTPUT field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 DATA** | RO | The output data for X-axis |

### 23.7.7 CORDIC_Y_OUTPUT (Y-axis Output Data Register)

Offset : 28h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CORDIC_Y_OUTPUT field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 DATA** | RO | The output data for Y-axis |

### 23.7.8 CORDIC_Z_OUTPUT (Z-axis Output Data Register)

Offset : 2Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CORDIC_Z_OUTPUT field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**DATA** | RO | The output data for Z-axis |

Flagchip

# Chapter 24 CPU System Control (CSC)

## 24.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • In the register section *CSC0_CCM0_CFG:*<br>- Updated the descriptions of the bit fields "CPU0_FORCE_HCLKEN" and "CCM0_HANDSHAKE_MODE";<br>- Changed the reset value for "CPU0_FORCE_HCLKEN" from "0b" to "1b".<br>• Removed the register section *CSC0_CCM0_STATUS;*<br>• In the register section *CSC0_PCU_CTRL:*<br>- Updated the descriptions of the bit fields "PAD_ISO_HOLD" and "PAD_ISO_HOLD_CLR";<br>- Changed the access attribute from "RW" to "WO" for "PAD_ISO_HOLD_CLR".<br>• In the register section *CSC0_SMU_CTRL0:*<br>- Changed the bit field "PMC_MON [30]" to "PMC_HVD_LVD [30];<br>- Changed the bit field "PMC_LVD_HVD [27]" to "PMC_DOUBLE_BOUND [27]";<br>- Changed the name of the bit field "MAM0_S2F_MON" to "MAM0_S5_S2F_MON".<br>• In the register section *CSC0_SMU_CTRL4:*<br>- Removed the bit fields "SCF_RST_EN [13]" and "SCF_IRQ_EN [12]";<br>- Add a new bit field "SMU_RIE [13:12]" |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## 24.2 Chip-specific Information

N/A

## 24.3 Features

- Each core has a dedicated CPU System Control (CSC) module
- Provide the low-power handshake control (by PCC) on peripherals
- Provide the corresponding CPU clock control status
- SCG MAM stall request for SCG clock source configuration or switch
- Provide the corresponding CPU software interrupt generation with CPU0_INT
- Selection of CLKOUT to be observed
- Selection of AON/32K clock source
- Provide control on the clock when the system enters low-power mode
- Provide control on the sequence to enter standby mode
- Provide the PAD isolation hold configuration
- Provide end-to-end ECC error-related control
- Provide decoder monitor error enable on the CPU TCM and system RAM
- Provide the SRAM0/1 Monitor and AFCB 0/1 monitor error enable to FCSMU
- Provide FCSMU/SCG/SCM CRC error enable
- Provide TMU error enable to FCSMU
- Provide CPU Overlay error enable to FCSMU
- Provide Matrix Access Monitor (MAM) timeout error enable
- Provide the CMU4, CMU2, CMU1 clock error enable to the FCSMU
- Provide CMU3 clock error enable control to the RGM
- Provide FLASH EDC error enable to FCSMU
- Provide the System RAM EDC, HSM IRAM, DRAM ECC error enable to FCMSU
- Provide DMA0_CFG ECC error and DMA0 Lockstep error enable to FCSMU
- Provide F0SC/PLL0/PLL1 loss of clock error enable to the FCSMU
- Provide the low power wakeup PAD configure source and polarity selection
- Support CSC0 Register Domain Write Permission
- Support CSC0 Register Lock function

## 24.4 Overview

The main function of the CPU System Control (CSC) module is for the chip-level miscellaneous control. For all detailed functions, refer to the corresponding chapters of different peripheral modules.

## 24.5 Functional Description

### 24.5.1 Peripherals Low Power Handshake

The CSC provides peripherals with low-power entry and exit with software stop requests.

### 24.5.2 CSC Register LOCK Function

For the LOCK bit, the user needs first to configure other bit fields except the LOCK bit, then configure the LOCK bit if needed. When the LOCK bit is set, other bit fields cannot be written again, and only a system reset can clear the LOCK bit.

### 24.5.3 CSC Register WPB_LOCK and WPB Function

For WPB_LOCK and WPB bits, the user needs first to configure the WPB bit to assign the write permission to CPU0 (WPB = 0 & 1, CPU0 can write), then configure the WPB_LOCK bit if needed. When the WPB_LOCK bit is set, the corresponding WPB bit

cannot be written again, but other bit fields can still be configured with the allowed CPU. And only the system reset to clear the WPB_LOCK bit.

## 24.6 CSC0 Register Definition

CSC0 base address : 4007_9000h

**Register Overview - CSC0**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | CSC0_STOP_MODER0 (STOP Mode Control Register 0) | 32 | RW | 0000_0000h | page |
| **4h** | CSC0_STOP_REQR0 (STOP Request Register 0) | 32 | RW | 0000_0000h | page |
| **8h** | CSC0_STOP_ACKR0 (STOP Acknowledge Register 0) | 32 | RO | FFFF_FFFFh | page |
| **Ch** | CSC0_STOP_MODER1 (STOP Mode Control Register 1) | 32 | RW | 0000_0000h | page |
| **10h** | CSC0_STOP_REQR1 (STOP Request Register 1) | 32 | RW | 0000_0000h | page |
| **14h** | CSC0_STOP_ACKR1 (STOP Acknowledge Register 1) | 32 | RO | FFFF_FFFFh | page |
| **18h** | CSC0_STOP_MODER2 (STOP Mode Control Register 2) | 32 | RW | 0000_0000h | page |
| **1Ch** | CSC0_STOP_REQR2 (STOP Request Register 2) | 32 | RW | 0000_0000h | page |
| **20h** | CSC0_STOP_ACKR2 (STOP Acknowledge Register 2) | 32 | RO | FFFF_FFFFh | page |
| **24h** | CSC0_CCM0_CFG (CCM0 Configuration Register) | 32 | RW | 0000_0004h | page |
| **2Ch** | CSC0_SCG_MAM_STALL (SCG MAM Stall Request Register) | 32 | RW | 0000_0000h | page |
| **30h** | CSC0_CPU0_INT (CPU0 Software Interrupt Register) | 32 | RW | 0000_0000h | page |
| **40h** | CSC0_CLKOUT_CTRL (Clkout Control Register) | 32 | RW | 0000_0000h | page |
| **44h** | CSC0_AONCLKSR (AON Clock Select Register) | 32 | RW | 0000_5300h | page |
| **48h** | CSC0_PCU_CTRL (PCU Control Register) | 32 | RW | 0800_0000h | page |
| **50h** | CSC0_SMU_CTRL0 (FCSMU Control Register 0) | 32 | RW | FBF6_9F6Eh | page |
| **54h** | CSC0_SMU_CTRL1 (FCSMU Control Register 1) | 32 | RW | 0007_F7FEh | page |
| **60h** | CSC0_SMU_CTRL4 (FCSMU Control Register 4) | 32 | RW | D333_F732h | page |
| **64h** | CSC0_CMU_CTRL (CMU Control Register) | 32 | RW | 0000_0000h | page |
| **68h** | CSC0_SMU_CTRL5 (FCSMU Control Register 5) | 32 | RW | 33F1_5714h | page |
| **70h** | CSC0_LP_WAKEUP (Low Power Wakeup Register) | 32 | RW | 0000_0000h | page |

### 24.6.1 CSC0_STOP_MODER0 (STOP Mode Control Register 0)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WPB_ | WPB | | | 0 | | | | | FLEX | FLEX | FREQ | 0 | | | FCUA |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | LOCK | | | | | | | | | CAN1 | CAN0 | M | | | | RT3 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | FCUART2 | FCUART1 | 0 | FCSPI2 | FCSPI1 | SENT0 | TMU | 0 | ADC1 | ADC0 | CMU4 | WDOG0 | 0 | | ISM0 | DMA0 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CSC0_STOP_MODER0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**WPB_LOCK** | RW | Write Permission Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**WPB** | RW | Write Permission<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |
| **22**<br>**FLEXCAN1** | RW | Control the stop acknowledge function enable or not to the FLEXCAN1.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **21**<br>**FLEXCAN0** | RW | Control the stop acknowledge function enable or not to the FLEXCAN0.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **20**<br>**FREQM** | RW | Control the stop acknowledge function enable or not to the FREQM.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **16**<br>**FCUART3** | RW | Control the stop acknowledge function enable or not to the FCUART3.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **15**<br>**FCUART2** | RW | Control the stop acknowledge function enable or not to the FCUART2.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |

| | | |
|---|---|---|
| **14**<br>**FCUART1** | RW | Control the stop acknowledge function enable or not to the FCUART1.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **12**<br>**FCSPI2** | RW | Control the stop acknowledge function enable or not to the FCSPI2.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **11**<br>**FCSPI1** | RW | Control the stop acknowledge function enable or not to the FCSPI1.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **10**<br>**SENT0** | RW | Control the stop acknowledge function enable or not to the SENT0.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **9**<br>**TMU** | RW | Control the stop acknowledge function enable or not to the TMU.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **7**<br>**ADC1** | RW | Control the stop acknowledge function enable or not to the ADC1.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **6**<br>**ADC0** | RW | Control the stop acknowledge function enable or not to the ADC0.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **5**<br>**CMU4** | RW | Control the stop acknowledge function enable or not to the CMU4.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **4**<br>**WDOG0** | RW | Control the stop acknowledge function enable or not to the WDOG0.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **1**<br>**ISM0** | RW | Control the stop acknowledge function enable or not to the ISM0.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **0**<br>**DMA0** | RW | Control the stop acknowledge function enable or not to the DMA0.<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |

### 24.6.2 CSC0_STOP_REQR0 (STOP Request Register 0)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WPB_ | WPB | | | 0 | | | | | FLEX | FLEX | FREQ | 0 | | | FCUA |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | LOCK | | | | | | | | | CAN1 | CAN0 | M | | | | RT3 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | FCUART2 | FCUART1 | 0 | FCSPI2 | FCSPI1 | SENT0 | TMU | 0 | ADC1 | ADC0 | CMU4 | WDOG0 | 0 | | ISM0 | DMA0 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CSC0_STOP_REQR0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**WPB_LOCK** | RW | Write Permission Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**WPB** | RW | Write Permission<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |
| **22**<br>**FLEXCAN1** | RW | Stop Request Set to FLEXCAN1<br>0b - Clear stop request<br>1b - Set stop request |
| **21**<br>**FLEXCAN0** | RW | Stop Request Set to FLEXCAN0<br>0b - Clear stop request<br>1b - Set stop request |
| **20**<br>**FREQM** | RW | Stop Request Set to FREQM<br>0b - Clear stop request<br>1b - Set stop request |
| **16**<br>**FCUART3** | RW | Stop Request Set to FCUART3<br>0b - Clear stop request<br>1b - Set stop request |
| **15**<br>**FCUART2** | RW | Stop Request Set to FCUART2<br>0b - Clear stop request<br>1b - Set stop request |

| | | |
|---|---|---|
| **14**<br>**FCUART1** | RW | Stop Request Set to FCUART1<br>0b - Clear stop request<br>1b - Set stop request |
| **12**<br>**FCSPI2** | RW | Stop Request Set to FCSPI2<br>0b - Clear stop request<br>1b - Set stop request |
| **11**<br>**FCSPI1** | RW | Stop Request Set to FCSPI1<br>0b - Clear stop request<br>1b - Set stop request |
| **10**<br>**SENT0** | RW | Stop Request Set to SENT0<br>0b - Clear stop request<br>1b - Set stop request |
| **9**<br>**TMU** | RW | Stop Request Set to TMU<br>0b - Clear stop request<br>1b - Set stop request |
| **7**<br>**ADC1** | RW | Stop Request Set to ADC1<br>0b - Clear stop request<br>1b - Set stop request |
| **6**<br>**ADC0** | RW | Stop Request Set to ADC0<br>0b - Clear stop request<br>1b - Set stop request |
| **5**<br>**CMU4** | RW | Stop Request Set to CMU4<br>0b - Clear stop request<br>1b - Set stop request |
| **4**<br>**WDOG0** | RW | Stop Request Set to WDOG0<br>0b - Clear stop request<br>1b - Set stop request |
| **1**<br>**ISM0** | RW | Stop Request Set to ISM0<br>0b - Clear stop request<br>1b - Set stop request |
| **0**<br>**DMA0** | RW | Stop Request Set to DMA0<br>0b - Clear stop request<br>1b - Set stop request |

### 24.6.3 CSC0_STOP_ACKR0 (STOP Acknowledge Register 0)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | FLEXCAN1 | FLEXCAN0 | FREQM | 0 | | | FCUART3 |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FCUART2 | FCUART1 | 0 | FCSPI2 | FCSPI1 | SENT0 | TMU | 0 | ADC1 | ADC0 | CMU4 | WDOG0 | 0 | | ISM0 | DMA0 |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**CSC0_STOP_ACKR0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **22**<br>**FLEXCAN1** | RO | FLEXCAN1 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **21**<br>**FLEXCAN0** | RO | FLEXCAN0 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **20**<br>**FREQM** | RO | FREQM Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **16**<br>**FCUART3** | RO | FCUART3 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **15**<br>**FCUART2** | RO | FCUART2 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **14**<br>**FCUART1** | RO | FCUART1 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **12**<br>**FCSPI2** | RO | FCSPI2 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |

| | | |
|---|---|---|
| **11**<br>**FCSPI1** | RO | FCSPI1 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **10**<br>**SENT0** | RO | SENT0 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **9**<br>**TMU** | RO | TMU Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **7**<br>**ADC1** | RO | ADC1 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **6**<br>**ADC0** | RO | ADC0 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **5**<br>**CMU4** | RO | CMU4 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **4**<br>**WDOG0** | RO | WDOG0 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **1**<br>**ISM0** | RO | ISM Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **0**<br>**DMA0** | RO | DMA0 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |

### 24.6.4 CSC0_STOP_MODER1 (STOP Mode Control Register 1)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WPB_<br>LOCK | WPB | | | 0 | | | FLEX<br>CAN3 | FLEX<br>CAN2 | 0 | MSC0 | 0 | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | FCUA | FCUA | FCUA | FCUA | FCII | 0 | FCSP | FCSP | FCSP | 0 | | | | WDOG | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | RT7 | RT6 | RT5 | RT4 | C1 | | I5 | I4 | I3 | | | | | 1 | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CSC0_STOP_MODER1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**WPB_LOCK** | RW | Write Permission Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**WPB** | RW | Write Permission<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |
| **24**<br>**FLEXCAN3** | RW | Control the stop acknowledge function enable or not to the FLEXCAN3<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **23**<br>**FLEXCAN2** | RW | Control the stop acknowledge function enable or not to the FLEXCAN2<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **21**<br>**MSC0** | RW | Control the stop acknowledge function enable or not to the MSC0<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **14**<br>**FCUART7** | RW | Control the stop acknowledge function enable or not to the FCUART7<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **13**<br>**FCUART6** | RW | Control the stop acknowledge function enable or not to the FCUART6<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **12**<br>**FCUART5** | RW | Control the stop acknowledge function enable or not to the FCUART5<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **11**<br>**FCUART4** | RW | Control the stop acknowledge function enable or not to the FCUART4<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |

| | | |
|---|---|---|
| **10**<br>**FCIIC1** | RW | Control the stop acknowledge function enable or not to the FCIIC1<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **8**<br>**FCSPI5** | RW | Control the stop acknowledge function enable or not to the FCSPI5<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **7**<br>**FCSPI4** | RW | Control the stop acknowledge function enable or not to the FCSPI4<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **6**<br>**FCSPI3** | RW | Control the stop acknowledge function enable or not to the FCSPI3<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **1**<br>**WDOG1** | RW | Control the stop acknowledge function enable or not to the WDOG1<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |

### 24.6.5 CSC0_STOP_REQR1 (STOP Request Register 1)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WPB_<br>LOCK | WPB | | | 0 | | | FLEX<br>CAN3 | FLEX<br>CAN2 | 0 | MSC0 | 0 | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | FCUA<br>RT7 | FCUA<br>RT6 | FCUA<br>RT5 | FCUA<br>RT4 | FCII<br>C1 | 0 | FCSP<br>I5 | FCSP<br>I4 | FCSP<br>I3 | 0 | | | | WDOG<br>1 | 0 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CSC0_STOP_REQR1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**WPB_LOCK** | RW | Write Permission Lock<br>0b - Not locked<br>1b - Locked |

| | | |
|---|---|---|
| **30-28**<br>**WPB** | RW | Write Permission<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |
| **24**<br>**FLEXCAN3** | RW | Stop Request Set to FLEXCAN3<br>0b - Clear stop request<br>1b - Set stop request |
| **23**<br>**FLEXCAN2** | RW | Stop Request Set to FLEXCAN2<br>0b - Clear stop request<br>1b - Set stop request |
| **21**<br>**MSC0** | RW | Stop Request Set to MSC0<br>0b - Clear stop request<br>1b - Set stop request |
| **14**<br>**FCUART7** | RW | Stop Request Set to FCUART7<br>0b - Clear stop request<br>1b - Set stop request |
| **13**<br>**FCUART6** | RW | Stop Request Set to FCUART6<br>0b - Clear stop request<br>1b - Set stop request |
| **12**<br>**FCUART5** | RW | Stop Request Set to FCUART5<br>0b - Clear stop request<br>1b - Set stop request |
| **11**<br>**FCUART4** | RW | Stop Request Set to FCUART4<br>0b - Clear stop request<br>1b - Set stop request |
| **10**<br>**FCIIC1** | RW | Stop Request Set to FCIIC1<br>0b - Clear stop request<br>1b - Set stop request |
| **8**<br>**FCSPI5** | RW | Stop Request Set to FCSPI5<br>0b - Clear stop request<br>1b - Set stop request |
| **7**<br>**FCSPI4** | RW | Stop Request Set to FCSPI4<br>0b - Clear stop request<br>1b - Set stop request |

| | | |
|---|---|---|
| **6**<br>**FCSPI3** | RW | Stop Request Set toFCSPI3<br>0b - Clear stop request<br>1b - Set stop request |
| **1**<br>**WDOG1** | RW | Stop Request Set to WDOG1<br>0b - Clear stop request<br>1b - Set stop request |

### 24.6.6 CSC0_STOP_ACKR1 (STOP Acknowledge Register 1)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | FLEXCAN3 | FLEXCAN2 | 0 | MSC0 | 0 | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | FCUART7 | FCUART6 | FCUART5 | FCUART4 | FCIIC1 | 0 | FCSPI5 | FCSPI4 | FCSPI3 | 0 | | | | WDOG1 | 0 |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**CSC0_STOP_ACKR1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **24**<br>**FLEXCAN3** | RO | FLEXCAN3 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **23**<br>**FLEXCAN2** | RO | FLEXCAN2 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **21**<br>**MSC0** | RO | MSC0 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **14**<br>**FCUART7** | RO | FCUART7 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |

| | | |
|---|---|---|
| **13**<br>**FCUART6** | RO | FCUART6 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **12**<br>**FCUART5** | RO | FCUART5 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **11**<br>**FCUART4** | RO | FCUART4 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **10**<br>**FCIIC1** | RO | FCIIC1 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **8**<br>**FCSPI5** | RO | FCSPI5 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **7**<br>**FCSPI4** | RO | FCSPI4 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **6**<br>**FCSPI3** | RO | FCSPI3 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **1**<br>**WDOG1** | RO | WDOG1 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |

### 24.6.7 CSC0_STOP_MODER2 (STOP Mode Control Register 2)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WPB_ | WPB | | | 0 | | | | | | | | | | | |
| W | LOCK | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | FCUA | FCII | FCSP | CMP1 | CMP0 | CMU3 | CMU2 | CMU1 | CMU0 | 0 | |
| W | | | | | | RT0 | C0 | I0 | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CSC0_STOP_MODER2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**WPB_LOCK** | RW | Write Permission Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**WPB** | RW | Write Permission<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |
| **10**<br>**FCUART0** | RW | Control the stop acknowledge function enable or not to the FCUART0<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **9**<br>**FCIIC0** | RW | Control the stop acknowledge function enable or not to the FCIIC0<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **8**<br>**FCSPI0** | RW | Control the stop acknowledge function enable or not to the FCSPI0<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **7**<br>**CMP1** | RW | Control the stop acknowledge function enable or not to the CMP1<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **6**<br>**CMP0** | RW | Control the stop acknowledge function enable or not to the CMP0<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **5**<br>**CMU3** | RW | Control the stop acknowledge function enable or not to the CMU3<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **4**<br>**CMU2** | RW | Control the stop acknowledge function enable or not to the CMU2<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |
| **3**<br>**CMU1** | RW | Control the stop acknowledge function enable or not to the CMU1<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |

| | | |
|---|---|---|
| **2**<br>**CMU0** | RW | Control the stop acknowledge function enable or not to the CMU0<br>0b - STOP_ACK disabled<br>1b - STOP_ACK enabled |

### 24.6.8 CSC0_STOP_REQR2 (STOP Request Register 2)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WPB_LOCK | WPB | | | 0 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | FCUART0 | FCIIC0 | FCSPI0 | CMP1 | CMP0 | CMU3 | CMU2 | CMU1 | CMU0 | 0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CSC0_STOP_REQR2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**WPB_LOCK** | RW | Write Permission Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**WPB** | RW | Write Permission<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |
| **10**<br>**FCUART0** | RW | Stop Request Set to FCUART0<br>0b - Clear stop request<br>1b - Set stop request |
| **9**<br>**FCIIC0** | RW | Stop Request Set to FCIIC0<br>0b - Clear stop request<br>1b - Set stop request |
| **8**<br>**FCSPI0** | RW | Stop Request Set to FCSPI0<br>0b - Clear stop request<br>1b - Set stop request |

| | | |
|---|---|---|
| **7**<br>**CMP1** | RW | Stop Request Set to CMP1<br>0b - Clear stop request<br>1b - Set stop request |
| **6**<br>**CMP0** | RW | Stop Request Set to CMP0<br>0b - Clear stop request<br>1b - Set stop request |
| **5**<br>**CMU3** | RW | Stop Request Set to CMU3<br>0b - Clear stop request<br>1b - Set stop request |
| **4**<br>**CMU2** | RW | Stop Request Set to CMU2<br>0b - Clear stop request<br>1b - Set stop request |
| **3**<br>**CMU1** | RW | Stop Request Set to CMU1<br>0b - Clear stop request<br>1b - Set stop request |
| **2**<br>**CMU0** | RW | Stop Request Set to CMU0<br>0b - Clear stop request<br>1b - Set stop request |

### 24.6.9 CSC0_STOP_ACKR2 (STOP Acknowledge Register 2)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | FCUART0 | FCIIC0 | FCSPI0 | CMP1 | CMP0 | CMU3 | CMU2 | CMU1 | CMU0 | 0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**CSC0_STOP_ACKR2 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **10**<br>**FCUART0** | RO | FCUART0 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **9**<br>**FCIIC0** | RO | FCIIC0 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **8**<br>**FCSPI0** | RO | FCSPI0 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **7**<br>**CMP1** | RO | CMP1 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **6**<br>**CMP0** | RO | CMP0 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **5**<br>**CMU3** | RO | CMU3 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **4**<br>**CMU2** | RO | CMU2 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **3**<br>**CMU1** | RO | CMU1 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |
| **2**<br>**CMU0** | RO | CMU0 Stop Acknowledge<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |

### 24.6.10 CSC0_CCM0_CFG (CCM0 Configuration Register)

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WPB_LOCK | WPB | | | 0 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | CPU0 | CCM0_HANDS | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | _FORCE_HCLKEN | HAKE_MODE | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**CSC0_CCM0_CFG field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**WPB_LOCK** | RW | Write Permission Lock<br>0b - Non is lock<br>1b - Lock |
| **30-28**<br>**WPB** | RW | Write Permission<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |
| **2**<br>**CPU0_FORCE_HCLKEN** | RW | CPU0 HCLOCK Enable<br>0b - Force CPU clock disable<br>1b - Force CPU clock enable |
| **1-0**<br>**CCM0_HANDSHAKE_MODE** | RW | CCM0 Handshake Mode<br>11b - Follow CPU stop acknowledge state<br>Others - Follow other masters' stop acknowledge state |

### 24.6.11 CSC0_SCG_MAM_STALL (SCG MAM Stall Request Register)

Offset : 2Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | SCG_STALL | LOCK |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CSC0_SCG_MAM_STALL field descriptions**

| Field | Type | Description |
|---|---|---|
| **1**<br>**SCG_STALL** | RW | SCG MAM Stall Request<br>NOTE: Need to assert the SCG_STALL to stall MAM when configuring the SCG clock source.<br>0b - SCG MAM stall request not asserted<br>1b - SCG MAM stall request asserts |
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Lock; can not be written |

### 24.6.12 CSC0_CPU0_INT (CPU0 Software Interrupt Register)

Offset : 30h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WPB_LOCK | WPB | | | 0 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | SW_INT |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CSC0_CPU0_INT field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**WPB_LOCK** | RW | Write Permission Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**WPB** | RW | Write Permission<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |

| | | |
|---|---|---|
| **0**<br>**SW_INT** | RW | CPU0 Software Interrupt Assert<br>0b - CPU0 software interrupt not asserted<br>1b - CPU0 software interrupt asserts |

### 24.6.13 CSC0_CLKOUT_CTRL (Clkout Control Register)

Offset : 40h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CLKOUT_EN | CLKOUT_DIV | | | CLKOUT_SEL | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | LOCK |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CSC0_CLKOUT_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**CLKOUT_EN** | RW | Enable Clkout<br>0b - Disable Clkout to pad<br>1b - Enable Clkout to pad |
| **30-28**<br>**CLKOUT_DIV** | RW | CLKOUT Divide Ratio<br>NOTE: The below sequence should be followed while CLKOUT configuration:<br>1. Disable CLKOUTEN<br>2. Set CLKOUTDIV<br>3. Enable CLKOUTEN<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 3<br>011b - Divide by 4<br>100b - Divide by 5<br>101b - Divide by 6<br>110b - Divide by 7<br>111b - Divide by 8 |

| | | |
|---|---|---|
| **27-24**<br>**CLKOUT_SEL** | RW | CLKOUT Select<br>NOTE: The below sequence should be followed while CLKOUT configuration:<br>1. Disable CLKOUTEN<br>2. Set CLKOUTSEL<br>3. Enable CLKOUTEN<br>0000b - SCG CLKOUT<br>0001b - Reserved<br>0010b - FOSC DIVM CLK<br>0011b - SLOW CLK<br>0100b - SIRC DIVM CLK<br>0101b - PLL1 DIVM CLK<br>0110b - FIRC DIVM CLK<br>0111b - CORE CLK<br>1000b - PLL0 DIVM CLK<br>1001b - BUS CLK<br>1010b - SIRC_128KHz CLK<br>1011b - Reserved<br>1100b - AON_CLK as selected by CSC0_AONCLKSR[AONCLKSEL]<br>1101b - Reserved<br>1110b - 32K_CLK as selected by CSC0_AONCLKSR[RTCCLKSEL]<br>1111b - Reserved |
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Lock; can not be written |

### 24.6.14 CSC0_AONCLKSR (AON Clock Select Register)

Offset : 44h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | AON32KCLKSEL | | RTCCLKSEL | | AONCLKSEL | | SIRC_DIV32KEN | AON1KCLKEN | 0 | | | | | | | LOCK |
| RST | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CSC0_AONCLKSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-14**<br>**AON32KCLKSEL** | RW | 32 KHz Always-on Clock Source Select<br>Selects 32 KHz Always on clock source for Pin filter/Stop ack clock.<br>00b - Reserved<br>01b - SIRCDIV_32K<br>10b - SOSC<br>11b - SIRC32K |
| **13-12**<br>**RTCCLKSEL** | RW | RTC Clock Source Select<br>Selects RTC clock source for peripherals.<br>00b - FOSCDIVL_CLK<br>01b - SIRCDIV_32K<br>10b - SOSC<br>11b - SIRC32K |
| **11-10**<br>**AONCLKSEL** | RW | AON Clock Source Select<br>Selects AON clock source for peripherals.<br>00b - SIRCDIV_128K<br>01b - SIRC32K<br>10b - SIRCDIV_32K<br>11b - SIRC32_1K |
| **9**<br>**SIRCDIV32KEN** | RW | SIRCDIV_32K Enable<br>0b - Disable SIRCDIV_32K output<br>1b - Enable SIRCDIV_32K output |
| **8**<br>**AON1KCLKEN** | RW | Always-on 1K Clock (SIRC32_1K) Enable<br>0b - Disable SIRC32_1K output<br>1b - Enable SIRC32_1K output |
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Lock; can not be written |

### 24.6.15 CSC0_PCU_CTRL (PCU Control Register)

Offset : 48h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RPM_EXIT_CNT | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | PAD_ | 0 | 0 | | | | | LOCK |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | ISO_HOLD | PAD_ISO_HOLD_CLR | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CSC0_PCU_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**RPM_EXIT_CNT** | RW | Count for Reduce Power Mode Exit |
| **7**<br>**PAD_ISO_HOLD** | RW | Power Domain 1 Isolation Enable Hold<br>0b - PD1 domain PAD state follow hardware control when enter and exit low power mode<br>1b - PD1 domain PAD state follow software control when enter and exit low power mode. After writing 1 to the PAD_ISO_HOLD_CLR bit, the PAD_ISO_HOLD bit will be cleared. |
| **6**<br>**PAD_ISO_HOLD_CLR** | WO | Power Domain 1 Isolation Enable Hold Clear<br>0b - Do not clear PD1 domain PAD state in software control mode<br>1b - Clear PD1 domain PAD state in software control mode |
| **0**<br>**LOCK** | RW | 0b - Unlock. can be written<br>1b - Lock; can not be written |

### 24.6.16 CSC0_SMU_CTRL0 (FCSMU Control Register 0)

Offset : 50h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CMU4 | PMC_HVD_LVD | CMU2 | CMU1 | PMC_DOUBLE_BOUND | 0 | SRAM1_MON | SRAM0_MON | CPM_MON | MAM0_S5_DS_ECC | MAM0_S8_DS | MAM0_S5_DS | 0 | DMA0 | HSM | 0 |
| RST | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAM0 | 0 | | CPU0 | MAM0 | MAM0 | MAM0 | MAM0 | 0 | MAM0 | MAM0 | 0 | MAM0 | MAM0 | MAM0 | LOCK |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | _S5_S2F_MON | | | _AHBS | _S8 | _S7 | _S6 | _S5 | | _S4 | _S3 | | _S2 | _S1 | _S0 | |
| RST | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

**CSC0_SMU_CTRL0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**CMU4** | RW | Enable CMU4 Clock Error (loss of clock error or clock mismatch error) to assert request to FCSMU.<br>0b - Disable CMU4 clock error (loss of clock error or clock mismatch error) to assert request to FCSMU.<br>1b - Enable CMU4 clock error (loss of clock error or clock mismatch error) to assert request to FCSMU. |
| **30**<br>**PMC_HVD_LVD** | RW | Enable PMC HVD/LVD Error Request to FCSMU<br>0b - Disable PMC HVD/LVD Error Request to FCSMU<br>1b - Enable PMC HVD/LVD Error Request to FCSMU |
| **29**<br>**CMU2** | RW | Enable CMU2 clock error (loss of clock error or clock mismatch error) to assert request to FCSMU.<br>0b - Disable CMU2 clock error (loss of clock error or clock mismatch error) to assert request to FCSMU.<br>1b - Enable CMU2 clock error (loss of clock error or clock mismatch error) to assert request to FCSMU. |
| **28**<br>**CMU1** | RW | Enable the CMU1 clock error (loss of clock error or clock mismatch error) to assert request to FCSMU.<br>0b - Disable the CMU1 clock error (loss of clock error or clock mismatch error) to assert request to FCSMU.<br>1b - Enable the CMU1 clock error (loss of clock error or clock mismatch error) to assert request to FCSMU. |
| **27**<br>**PMC_DOUBLE_BOUND** | RW | PMC V11/V25 Double Bound Error to FCSMU<br>0b - Disable PMC V11/V25 Double Bound Error to FCSMU<br>1b - Enable PMC V11/V25 Double Bound Error to FCSMU |
| **25**<br>**SRAM1_MON** | RW | SRAM1 Decoder Monitor Error Enable to FCSMU<br>0b - Disable SRAM1 Decoder Monitor Error to FCSMU<br>1b - Enable SRAM1 Decoder Monitor Error to FCSMU |
| **24**<br>**SRAM0_MON** | RW | SRAM0 Decoder Monitor Error Enable to FCSMU<br>0b - Disable SRAM0 Decoder Monitor Error to FCSMU<br>1b - Enable SRAM0 Decoder Monitor Error to FCSMU |

| | | |
|---|---|---|
| **23**<br>**CPM_MON** | RW | Enable the CPU Private Module (CPM) monitor error to assert request to FCSMU.<br>0b - Disable the CPM monitor error to FCSMU<br>1b - Enable the CPM monitor error to FCSMU |
| **22**<br>**MAM0_S5_DS_ECC** | RW | MAM0 S5 Downsize ECC Error Enable to FCSMU<br>Including MAM0_S5_DS_ECC Single Bit Data Error, Multi Bits Data Error, and Address Error.<br>0b - Disable MAM0 S5 Downsize ECC Error to FCSMU<br>1b - Enable MAM S5 Downsize ECC Error to FCSMU |
| **21**<br>**MAM0_S8_DS** | RW | MAM0 S8 Downsize Monitor Error Enable to FCSMU<br>0b - Disable MAM0 S8 Downsize Monitor Eror to FCSMU<br>1b - Enable MAM0 S8 Downsize Monitor Eror to FCSMU |
| **20**<br>**MAM0_S5_DS** | RW | MAM0 S5 Downsize Monitor Enable to FCSMU<br>0b - Disable MAM0 S5 Downsize Monitor to FCSMU<br>1b - Enable MAM0 S5 Downsize Monitor to FCSMU |
| **18**<br>**DMA0** | RW | DMA0 ECC Error Enable to FCSMU<br>Including DMA0_ECC Single Bit Data Error, Multi Bits Data Error, and Address Error.<br>0b - Disable DMA0 ECC Error to FCSMU<br>1b - Enable DMA0 ECC Error to FCSMU |
| **17**<br>**HSM** | RW | HSM ECC Error Enable to FCSMU<br>Including HSM_ECC Single Bit Data Error, Multi Bits Data Error, and Address Error.<br>0b - Disable HSM ECC Error to FCSMU<br>1b - Enable HSM ECC Error to FCSMU |
| **15**<br>**MAM0_S5_S2F_MON** | RW | MAM0 S5 Slow to Fast AHBS Monitor Error Enable to FCSMU<br>0b - Disable MAM0 S5 Slow to Fast AHBS Monitor Error to FCSMU<br>1b - Enable MAM0 S5 Slow to Fast AHBS Monitor Error to FCSMU |
| **12**<br>**CPU0_AHBS** | RW | CPU0 AHBS ECC Error Enable to FCSMU<br>Including CPU0_AHBS_ECC Single Bit Data Error, Multi Bits Data Error, and Address Error.<br>0b - Disable CPU0 AHBS ECC Error to FCSMU<br>1b - Enable CPU0 AHBS ECC Error to FCSMU |
| **11**<br>**MAM0_S8** | RW | MAM0 S8 ECC Error Enable to FCSMU<br>Including MAM0_S8_ECC Single Bit Data Error, Multi Bits Data Error, Address Error, and Attribute Error.<br>0b - Disable MAM0 S8 ECC Error to FCSMU<br>1b - Enable MAM0 S8 ECC Error to FCSMU |

| | | |
|---|---|---|
| **10**<br>**MAM0_S7** | RW | MAM0 S7 ECC Error Enable to FCSMU<br>Including MAM0_S7_ECC Single Bit Data Error, Multi Bits Data Error, Address Error, and Attribute Error.<br>0b - Disable MAM0 S7 ECC Error to FCSMU<br>1b - Enable MAM0 S7 ECC Error to FCSMU |
| **9**<br>**MAM0_S6** | RW | MAM0 S6 ECC Error Enable to FCSMU<br>Including MAM0_S6_ECC Single Bit Data Error, Multi Bits Data Error, Address Error, and Attribute Error.<br>0b - Disable MAM0 S6 ECC Error to FCSMU<br>1b - Enable MAM0 S6 ECC Error to FCSMU |
| **8**<br>**MAM0_S5** | RW | MAM0 S5 ECC Error Enable to FCSMU<br>Including MAM0_S5_ECC Single Bit Data Error, Multi Bits Data Error, Address Error, and Attribute Error.<br>0b - Disable MAM0 S5 ECC Error to FCSMU<br>1b - Enable MAM0 S5 ECC Error to FCSMU |
| **6**<br>**MAM0_S4** | RW | MAM0 S4 ECC Error Enable to FCSMU<br>Including MAM0_S4_ECC Single Bit Data Error, Multi Bits Data Error, Address Error, and Attribute Error.<br>0b - Disable MAM0 S4 ECC Error to FCSMU<br>1b - Enable MAM0 S4 ECC Error to FCSMU |
| **5**<br>**MAM0_S3** | RW | MAM0 S3 ECC Error Enable to FCSMU<br>Including MAM0_S3_ECC Single Bit Data Error, Multi Bits Data Error, Address Error, and Attribute Error.<br>0b - Disable MAM0 S3 ECC Error to FCSMU<br>1b - Enable MAM0 S3 ECC Error to FCSMU |
| **3**<br>**MAM0_S2** | RW | MAM0 S2 ECC Error Enable to FCSMU<br>Including MAM0_S2_ECC Single Bit Data Error, Multi Bits Data Error, Address Error, and Attribute Error.<br>0b - Disable MAM0 S2 ECC Error to FCSMU<br>1b - Enable MAM0 S2 ECC Error to FCSMU |
| **2**<br>**MAM0_S1** | RW | MAM0 S1 ECC Error Enable to FCSMU<br>Including MAM0_S1_ECC Single Bit Data Error, Multi Bits Data Error, Address Error, and Attribute Error.<br>0b - Disable MAM0 S1 ECC Error to FCSMU<br>1b - Enable MAM0 S1 ECC Error to FCSMU |
| **1**<br>**MAM0_S0** | RW | MAM0 S0 ECC Error Enable to FCSMU<br>Including MAM0_S0_ECC Single Bit Data Error, Multi Bits Data Error, Address Error, and Attribute Error.<br>0b - Disable MAM0 S0 ECC Error to FCSMU<br>1b - Enable MAM0 S0 ECC Error to FCSMU |

| | | |
|---|---|---|
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Lock; can not be written |

### 24.6.17 CSC0_SMU_CTRL1 (FCSMU Control Register 1)

Offset : 54h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | CPU0_DTCM1_S_EN | CPU0_DTCM0_S_EN | CPU0_ITCM_S_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CPU0_OVERLAY | CPU0_DTCM1_MON | CPU0_DTCM0_MON | CPU0_ITCM_MON | 0 | CPU0_AHBM1_F2S | CPU0_AHBM0_F2S | CPU0_AHBP_F2S | CPU0_DTCM1 | CPU0_DTCM0 | CPU0_ITCM | CPU0_DCACHE | CPU0_ICACHE | CPU0_AHBP | CPU0_AHBM | LOCK |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

**CSC0_SMU_CTRL1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **18**<br>**CPU0_DTCM1_S_EN** | RW | CPU0 DTCM1 Single Bit Data Error Enable to FCSMU<br>0b - Disable CPU0 DTCM1 Single Bit Data Error to FCSMU<br>1b - Enable CPU0 DTCM1 Single Bit Data Error to FCSMU |
| **17**<br>**CPU0_DTCM0_S_EN** | RW | CPU0 DTCM0 Single Bit Data Error Enable to FCSMU<br>0b - Disable CPU0 DTCM0 Single Bit Data Error to FCSMU<br>1b - Enable CPU0 DTCM0 Single Bit Data Error to FCSMU |
| **16**<br>**CPU0_ITCM_S_EN** | RW | CPU0 ITCM Single Bit Data Error Enable to FCSMU<br>0b - Disable CPU0 ITCM Single Bit Data Error to FCSMU<br>1b - Enable CPU0 ITCM Single Bit Data Error to FCSMU |
| **15**<br>**CPU0_OVERLAY** | RW | CPU0 Overlay Error Enable to FCSMU<br>0b - Disable CPU0 Overlay Error to FCSMU<br>1b - Enable CPU0 Overlay Error to FCSMU |

| | | |
|---|---|---|
| **14**<br>**CPU0_DTCM1_MON** | RW | CPU0 DTCM1 Decoder Monitor Error Enable to FCSMU<br>0b - Disable CPU0 DTCM1 Decoder Monitor Error to FCSMU<br>1b - Enable CPU0 DTCM1 Decoder Monitor Error to FCSMU |
| **13**<br>**CPU0_DTCM0_MON** | RW | CPU0 DTCM0 Decoder Monitor Error Enable to FCSMU<br>0b - Disable CPU0 DTCM0 Decoder Monitor Error to FCSMU<br>1b - Enable CPU0 DTCM0 Decoder Monitor Error to FCSMU |
| **12**<br>**CPU0_ITCM_MON** | RW | CPU0 ITCM Decoder Monitor Error Enable to FCSMU<br>0b - Disable CPU0 ITCM Decoder Monitor Error to FCSMU<br>1b - Enable CPU0 ITCM Decoder Monitor Error to FCSMU |
| **10**<br>**CPU0_AHBM1_F2S** | RW | CPU0 AHBM1 Fast to Slow Monitor Error Enable to FCSMU<br>0b - Disable CPU0 AHBM1 Fast to Slow Monitor Error to FCSMU<br>1b - Enable CPU0 AHBM1 Fast to Slow Monitor Error to FCSMU |
| **9**<br>**CPU0_AHBM0_F2S** | RW | CPU0 AHBM0 Fast to Slow Monitor Error Enable to FCSMU<br>0b - Disable CPU0 AHBM0 Fast to Slow Monitor Error to FCSMU<br>1b - Enable CPU0 AHBM0 Fast to Slow Monitor Error to FCSMU |
| **8**<br>**CPU0_AHBP_F2S** | RW | CPU0 AHBP Fast to Slow Monitor Error Enable to FCSMU<br>0b - Disable CPU0 AHBP Fast to Slow Monitor Error to FCSMU<br>1b - Enable CPU0 AHBP Fast to Slow Monitor Error to FCSMU |
| **7**<br>**CPU0_DTCM1** | RW | CPU0 DTCM1 ECC Error Enable to FCSMU<br>Including CPU0 DTCM1 ECC Single Bit Data Error, Multi Bits Data Error, and Address Error.<br>0b - Disable CPU0 DTCM1 ECC Error to FCSMU<br>1b - Enable CPU0 DTCM1 ECC Error to FCSMU |
| **6**<br>**CPU0_DTCM0** | RW | CPU0 DTCM0 ECC Error Enable to FCSMU<br>Including CPU0 DTCM0 ECC Single Bit Data Error, Multi Bits Data Error, and Address Error<br>0b - Disable CPU0 DTCM0 ECC Error to FCSMU<br>1b - Enable CPU0 DTCM0 ECC Error to FCSMU |
| **5**<br>**CPU0_ITCM** | RW | CPU0 ITCM ECC Error Enable to FCSMU<br>Including CPU0 ITCM ECC Single Bit Data Error, Multi Bits Data Error, and Address Error.<br>0b - Disable CPU0 ITCM ECC Error to FCSMU<br>1b - Enable CPU0 ITCM ECC Error to FCSMU |
| **4**<br>**CPU0_DCACHE** | RW | CPU0 Data CACHE Error Enable to FCSMU<br>Including CPU0 DCACHE DATA0 Error, DATA1 Error, and TAG Error.<br>0b - Disable CPU0 Data CACHE Error to FCSMU<br>1b - Enable CPU0 Data CACHE Error to FCSMU |

| | | |
|---|---|---|
| **3**<br>**CPU0_ICACHE** | RW | CPU0 Code CACHE Error Enable to FCSMU<br>Including CPU0 ICACHE DATA0 Error, DATA1 Error, and TAG Error.<br>0b - Disable CPU0 Code CACHE Error to FCSMU<br>1b - Enable CPU0 Code CACHE Error to FCSMU |
| **2**<br>**CPU0_AHBP** | RW | CPU0 AHBP ECC Error Enable to FCSMU<br>Including CPU0_AHBP_ECC Single Bit Data Error, Multi Bits Data Error, and Address Error.<br>0b - Disable CPU0 AHBP ECC Error to FCSMU<br>1b - Enable CPU0 AHBP ECC Error to FCSMU |
| **1**<br>**CPU0_AHBM** | RW | CPU0 AHBM ECC Error Enable to FCSMU<br>Including CPU0_AHBM_ECC Single Bit Data Error, Multi Bits Data Error, and Address Error.<br>0b - Disable CPU0 AHBM ECC Error to FCSMU<br>1b - Enable CPU0 AHBM ECC Error to FCSMU |
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Lock; can not be written |

### 24.6.18 CSC0_SMU_CTRL4 (FCSMU Control Register 4)

Offset : 60h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HSM_WDOG | FMC_ERR | 0 | ROM_S_EN | 0 | | SRAM1_EDC_MON_EN | SRAM0_EDC_MON_EN | 0 | | SRAM1_S_EN | SRAM0_S_EN | 0 | | FLASH_ECC_EN_S | FLASH_ECC_EN |
| RST | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | AFCB1_MON | AFCB0_MON | SMU_RIE | | 0 | TMU | SCM_CRC | SCG_CRC | 0 | | SRAM1_ECC | SRAM0_ECC | 0 | | MAM0_ERR | LOCK |
| RST | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |

**CSC0_SMU_CTRL4 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**HSM_WDOG** | RW | HSM WDOG Request Enable to FCSMU<br>0b - Disable HSM WDOG Request to FCSMU<br>1b - Enable HSM WDOG Request to FCSMU |

| | | |
|---|---|---|
| **30**<br>**FMC_ERR** | RW | FMC ECC Error Enable to FCSMU<br>Including FCCU Alarm Linebuffer Self Single Bit Error, and FCCU Alarm Linebuffer Self Multi Bit Error.<br>0b - Disable FMC ECC Error to FCSMU<br>1b - Enable FMC ECC Error to FCSMU |
| **28**<br>**ROM_S_EN** | RW | ROM Single Bit Data Error Enable to FCSMU<br>0b - Disable ROM Single Bit Data Error to FCSMU<br>1b - Enable ROM Single Bit Data Error to FCSMU |
| **25**<br>**SRAM1_EDC_MON_EN** | RW | SRAM1 EDC Monitor Error Enable to FCSMU<br>0b - Disable SRAM1 EDC Monitor Error to FCSMU<br>1b - Enable SRAM1 EDC Monitor Error to FCSMU |
| **24**<br>**SRAM0_EDC_MON_EN** | RW | SRAM0 EDC Monitor Error Enable to FCSMU<br>0b - Disable SRAM0 EDC Monitor Error to FCSMU<br>1b - Enable SRAM0 EDC Monitor Error to FCSMU |
| **21**<br>**SRAM1_S_EN** | RW | SRAM1 Single Bit Data Error Enable to FCSMU<br>0b - Disable SRAM1 Single Bit Data Error to FCSMU<br>1b - Enable SRAM1 Single Bit Data Error to FCSMU |
| **20**<br>**SRAM0_S_EN** | RW | SRAM0 Single Bit Data Error Enable to FCSMU<br>0b - Disable SRAM0 Single Bit Data Error to FCSMU<br>1b - Enable SRAM0 Single Bit Data Error to FCSMU |
| **17**<br>**FLASH_ECC_EN_S** | RW | FLASH Single Bit Error Enable to FCSMU<br>Including FLASH Single Bit Code Error, and Single Bit Data Error<br>0b - Disable FLASH Single Bit Error to FCSMU<br>1b - Enable FLASH Single Bit Error to FCSMU |
| **16**<br>**FLASH_ECC_EN** | RW | FLASH ECC Error Enable to FCSMU<br>Including FLASH Single Bit Code Error, Single Bit Data Error, Multi Bit Code Error, and Multi Bit Data Error.<br>0b - Disable FLASH ECC Error to FCSMU<br>1b - Enable FLASH ECC Error to FCSMU |
| **15**<br>**AFCB1_MON** | RW | AFCB1 Monitor Error Enable to FCSMU<br>0b - Disable AFCB1 Monitor Error to FCSMU<br>1b - Enable AFCB1 Monitor Error to FCSMU |
| **14**<br>**AFCB0_MON** | RW | AFCB0 Monitor Error Enable to FCSMU<br>0b - Disable AFCB0 Monitor Error to FCSMU<br>1b - Enable AFCB0 Monitor Error to FCSMU |

| | | |
|---|---|---|
| **13-12**<br>**SMU_RIE** | RW | FCSMU CRC Error Interrupt or Reset Request Enable<br>00b - Disable FCSMU CRC Error to assert the reset request to RGM or assert the interrupt request.<br>01b - Enable FCSMU CRC Error to assert the interrupt request.<br>10b - Enable FCSMU CRC Error to assert the reset request to RGM.<br>11b - Enable FCSMU CRC Error to assert the reset request to RGM. |
| **10**<br>**TMU** | RW | TMU Error Enable to FCSMU<br>0b - Disable TMU Error to FCSMU<br>1b - Enable TMU Error to FCSMU |
| **9**<br>**SCM_CRC** | RW | SCM CRC Error Enable to FCSMU<br>0b - Disable SCM CRC Error to FCSMU<br>1b - Enable SCM CRC Error to FCSMU |
| **8**<br>**SCG_CRC** | RW | SCG CRC Error Enable to FCSMU<br>0b - Disable SCG CRC Error to FCSMU<br>1b - Enable SCG CRC Error to FCSMU |
| **5**<br>**SRAM1_ECC** | RW | SRAM1 ECC Error Enable to FCSMU<br>Including SRAM1_ECC Single Bit Data Error, Multi Bits Data Error, and Address Error.<br>0b - Disable SRAM1 ECC Error to FCSMU<br>1b - Enable SRAM1 ECC Error to FCSMU |
| **4**<br>**SRAM0_ECC** | RW | SRAM0 ECC Error Enable to FCSMU<br>Including SRAM0_ECC Single Bit Data Error, Multi Bits Data Error, and Address Error.<br>0b - Disable SRAM0 ECC Error to FCSMU<br>1b - Enable SRAM0 ECC Error to FCSMU |
| **1**<br>**MAM0_ERR** | RW | Matrix Access Monitor 0 Timeout Error Enable to FCSMU<br>0b - Disable Matrix Access Monitor 0 Timeout Error to FCSMU<br>1b - Enable Matrix Access Monitor 0 Timeout Error to FCSMU |
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Lock; can not be written |

### 24.6.19 CSC0_CMU_CTRL (CMU Control Register)

Offset : 64h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | DMA0_CFG_S_EN_ADDR |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | CMU3_FCSMU_RST | CMU3_RST_EN | 0 | | CMU4_LOC_EN | 0 | | | | LOCK |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CSC0_CMU_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **16**<br>**DMA0_CFG_S_EN_ADDR** | RW | Enable to Report Single Error When One Bit Address_ECC Error Inject, Otherwise Report Address Error<br>0b - Report Address Error When One Bit Address_ECC Error Inject<br>1b - Report Single Error When One Bit Address_ECC Error Inject |
| **9**<br>**CMU3_FCSMU_RST** | RW | Enable CMU3 clock error to assert the reset request to RGM to reset System and FCSMU with CMU3_RST_EN set<br>0b - Disable CMU3 clock error to assert the reset request to RGM to reset System and FCSMU with CMU3_RST_EN set<br>1b - Enable CMU3 clock error to assert the reset request to RGM to reset System and FCSMU with CMU3_RST_EN set |
| **8**<br>**CMU3_RST_EN** | RW | Enable CMU3 clock error to assert the reset request to RGM to reset System<br>0b - Disable CMU3 clock error to assert the reset request to RGM to reset System<br>1b - Enable CMU3 clock error to assert the reset request to RGM to reset System |
| **5**<br>**CMU4_LOC_EN** | RW | Enable CMU4 Loss of clock error request to SCG<br>0b - Disable CMU4 Loss of clock error request to SCG<br>1b - Enable CMU4 Loss of clock error request to SCG |
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Lock; can not be written |

### 24.6.20 CSC0_SMU_CTRL5 (FCSMU Control Register 5)

Offset : 68h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | SRAM | SRAM | 0 | | FLAS | FLAS | HSM_ | HSM_ | HSM_ | HSM_ | 0 | | | DMA0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | 1_EDC | 0_EDC | | | H_EDC_P1_EN | H_EDC_P0_EN | IRAM_S | DRAM_S | IRAM | DRAM | | | | _ECC_S |
| RST | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | DMA0_ECC | 0 | DMA0_LOCKSTEP | 0 | FOSC_ERR | PLL1_ERR | PLL0_ERR | 0 | | | STCU_ERR | 0 | NONUSER_ERR | NVR_ERR | LOCK |
| RST | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

**CSC0_SMU_CTRL5 field descriptions**

| Field | Type | Description |
|---|---|---|
| **29**<br>**SRAM1_EDC** | RW | SRAM1 EDC Error Enable to FCSMU<br>0b - Disable SRAM1 EDC Error to FCSMU<br>1b - Enable SRAM1 EDC Error to FCSMU |
| **28**<br>**SRAM0_EDC** | RW | SRAM0 EDC Error Enable to FCSMU<br>0b - Disable SRAM0 EDC Error to FCSMU<br>1b - Enable SRAM0 EDC Error to FCSMU |
| **25**<br>**FLASH_EDC_P1_EN** | RW | FLASH P1 EDC Error Enable to FCSMU<br>0b - Disable FLASH P1 EDC to FCSMU<br>1b - Enable FLASH P1 EDC to FCSMU |
| **24**<br>**FLASH_EDC_P0_EN** | RW | FLASH P0 EDC Error Enable to FCSMU<br>0b - Disable FLASH P0 EDC to FCSMU<br>1b - Enable FLASH P0 EDC to FCSMU |
| **23**<br>**HSM_IRAM_S** | RW | HSM IRAM Single Bit Data Error Enable to FCSMU<br>0b - Disable HSM IRAM Single Bit Data Error to FCSMU<br>1b - Enable HSM IRAM Single Bit Data Error to FCSMU |
| **22**<br>**HSM_DRAM_S** | RW | HSM DRAM Single Bit Data Error Enable to FCSMU<br>0b - Disable HSM DRAM Single Bit Data Error to FCSMU<br>1b - Enable HSM DRAM Single Bit Data Error to FCSMU |
| **21**<br>**HSM_IRAM** | RW | HSM IRAM ECC Error Enable to FCSMU<br>Including HSM_IRAM_ECC Single Bit Data Error, Multi Bits Data Error, and Address Error.<br>0b - Disable HSM IRAM ECC Error to FCSMU<br>1b - Enable HSM IRAM ECC Error to FCSMU |

| | | |
|---|---|---|
| **20**<br>**HSM_DRAM** | RW | HSM DRAM ECC Error Enable to FCSMU<br>Including HSM_DRAM_ECC Single Bit Data Error, Multi Bits Data Error, and Address Error.<br>0b - Disable HSM DRAM ECC Error to FCSMU<br>1b - Enable HSM DRAM ECC Error to FCSMU |
| **16**<br>**DMA0_ECC_S** | RW | DMA0_CFG_ECC Single Bit Data Error Enable to FCSMU<br>0b - Disable DMA0 Single Bit Data Error to FCSMU<br>1b - Enable DMA0 Single Bit Data Error to FCSMU |
| **14**<br>**DMA0_ECC** | RW | DMA0 CFG ECC Error Enable to FCSMU<br>Including DMA0_CFG_ECC Single Bit Data Error, Multi Bits Data Error, and Address Error.<br>0b - Disable DMA0 ECC Error to FCSMU<br>1b - Enable DMA0 ECC Error to FCSMU |
| **12**<br>**DMA0_LOCKSTEP** | RW | DMA0 Lockstep Error Enable to FCSMU<br>0b - Disable DMA0 Lockstep Error to FCSMU<br>1b - Enable DMA0 Lockstep Error to FCSMU |
| **10**<br>**FOSC_ERR** | RW | FOSC Loss of Clock Error Enable to FCSMU<br>0b - Disable FOSC Loss of Clock Error to FCSMU<br>1b - Enable FOSC Loss of Clock Error to FCSMU |
| **9**<br>**PLL1_ERR** | RW | PLL1 Loss of Clock Error Enable to FCSMU<br>0b - Disable PLL1 Loss of Clock Error to FCSMU<br>1b - Enable PLL1 Loss of Clock Error to FCSMU |
| **8**<br>**PLL0_ERR** | RW | PLL0 Loss of Clock Error Enable to FCSMU<br>0b - Disable PLL0 Loss of Clock Error to FCSMU<br>1b - Enable PLL0 Loss of Clock Error to FCSMU |
| **4**<br>**STCU_ERR** | RW | STCU Self Test Error Enable to FCSMU<br>0b - Disable STCU Self Test Error to FCSMU<br>1b - Enable STCU Self Test Error to FCSMU |
| **2**<br>**NONUSER_ERR** | RW | STCU Non User Error Enable to FCSMU<br>0b - Disable STCU Non User Error to FCSMU<br>1b - Enable STCU Non User Error to FCSMU |
| **1**<br>**NVR_ERR** | RW | NVR Error Enable During Load to FCSMU<br>0b - Disable NVR Error to FCSMU<br>1b - Enable NVR Error to FCSMU |
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Lock; can not be written |

### 24.6.21 CSC0_LP_WAKEUP (Low Power Wakeup Register)

Offset : 70h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LP_WAKEUP_CFG0 | | | | LP_WAKEUP_CFG1 | | | | LP_WAKEUP_CFG2 | | | | LP_WAKEUP_CFG3 | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LP_WAKEUP_CFG4 | | | | 0 | | | | LP0_POL | LP1_POL | LP2_POL | LP3_POL | LP4_POL | 0 | | LOCK |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CSC0_LP_WAKEUP field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-28**<br>**LP_WAKEUP_CFG0** | RW | LP_WAKEUP_PAD0 Configure Source<br>0000b - Disable<br>0001b - AONTIMER Trigger<br>0010b - RTC Time Alarm<br>0011b - CMP0 Output<br>0100b - CMP1 Output<br>0101b - Reserved<br>0110b - PMC RPM Entry<br>0111b - WKU Interrupt<br>1000b - TSTMP0 PWM TRIGGER0, use modulate 0 as posedge, modulate 1 as negedge<br>1001b - TSTMP0 PWM TRIGGER1, use modulate 2 as posedge, modulate 3 as negedge<br>1010b - FCPT0 PWM TRIGGER0, use ch0 trigger as posedge, ch1 trigger as negedge<br>1011b - FCPT0 PWM TRIGGER1, use ch2 trigger as posedge, ch3 trigger as negedge<br>1100b - FCPIT0 Trigger out[0]<br>1101b - FCPIT0 Trigger out[1]<br>1110b - FCPIT0 Trigger out[2]<br>1111b - FCPIT0 Trigger out[3] |

| | | |
|---|---|---|
| **27-24**<br>**LP_WAKEUP_CFG1** | RW | LP_WAKEUP_PAD1 Configure Source<br>0000b - Disable<br>0001b - AONTIMER Trigger<br>0010b - RTC Time Alarm<br>0011b - CMP0 Output<br>0100b - CMP1 Output<br>0101b - Reserved<br>0110b - PMC RPM Entry<br>0111b - WKU Interrupt<br>1000b - TSTMP0 PWM TRIGGER0, use modulate 0 as posedge, modulate 1 as negedge<br>1001b - TSTMP0 PWM TRIGGER1, use modulate 2 as posedge, modulate 3 as negedge<br>1010b - FCPT0 PWM TRIGGER0, use ch0 trigger as posedge, ch1 trigger as negedge<br>1011b - FCPT0 PWM TRIGGER1, use ch2 trigger as posedge, ch3 trigger as negedge<br>1100b - FCPIT0 Trigger out[0]<br>1101b - FCPIT0 Trigger out[1]<br>1110b - FCPIT0 Trigger out[2]<br>1111b - FCPIT0 Trigger out[3] |
| **23-20**<br>**LP_WAKEUP_CFG2** | RW | LP_WAKEUP_PAD2 Configure Source<br>0000b - Disable<br>0001b - AONTIMER Trigger<br>0010b - RTC Time Alarm<br>0011b - CMP0 Output<br>0100b - CMP1 Output<br>0101b - Reserved<br>0110b - PMC RPM Entry<br>0111b - WKU Interrupt<br>1000b - TSTMP0 PWM TRIGGER0, use modulate 0 as posedge, modulate 1 as negedge<br>1001b - TSTMP0 PWM TRIGGER1, use modulate 2 as posedge, modulate 3 as negedge<br>1010b - FCPT0 PWM TRIGGER0, use ch0 trigger as posedge, ch1 trigger as negedge<br>1011b - FCPT0 PWM TRIGGER1, use ch2 trigger as posedge, ch3 trigger as negedge<br>1100b - FCPIT0 Trigger out[0]<br>1101b - FCPIT0 Trigger out[1]<br>1110b - FCPIT0 Trigger out[2]<br>1111b - FCPIT0 Trigger out[3] |

| | | |
|---|---|---|
| **19-16**<br>**LP_WAKEUP_CFG3** | RW | LP_WAKEUP_PAD3 Configure Source<br>0000b - Disable<br>0001b - AONTIMER Trigger<br>0010b - RTC Time Alarm<br>0011b - CMP0 Output<br>0100b - CMP1 Output<br>0101b - Reserved<br>0110b - PMC RPM Entry<br>0111b - WKU Interrupt<br>1000b - TSTMP0 PWM TRIGGER0, use modulate 0 as posedge, modulate 1 as negedge<br>1001b - TSTMP0 PWM TRIGGER1, use modulate 2 as posedge, modulate 3 as negedge<br>1010b - FCPT0 PWM TRIGGER0, use ch0 trigger as posedge, ch1 trigger as negedge<br>1011b - FCPT0 PWM TRIGGER1, use ch2 trigger as posedge, ch3 trigger as negedge<br>1100b - FCPIT0 Trigger out[0]<br>1101b - FCPIT0 Trigger out[1]<br>1110b - FCPIT0 Trigger out[2]<br>1111b - FCPIT0 Trigger out[3] |
| **15-12**<br>**LP_WAKEUP_CFG4** | RW | LP_WAKEUP_PAD4 Configure Source<br>0000b - Disable<br>0001b - AONTIMER Trigger<br>0010b - RTC Time Alarm<br>0011b - CMP0 Output<br>0100b - CMP1 Output<br>0101b - Reserved<br>0110b - PMC RPM Entry<br>0111b - WKU Interrupt<br>1000b - TSTMP0 PWM TRIGGER0, use modulate 0 as posedge, modulate 1 as negedge<br>1001b - TSTMP0 PWM TRIGGER1, use modulate 2 as posedge, modulate 3 as negedge<br>1010b - FCPT0 PWM TRIGGER0, use ch0 trigger as posedge, ch1 trigger as negedge<br>1011b - FCPT0 PWM TRIGGER1, use ch2 trigger as posedge, ch3 trigger as negedge<br>1100b - FCPIT0 Trigger out[0]<br>1101b - FCPIT0 Trigger out[1]<br>1110b - FCPIT0 Trigger out[2]<br>1111b - FCPIT0 Trigger out[3] |

| | | |
|---|---|---|
| **7**<br>**LP0_POL** | RW | LP_WAKEUP_CFG0 Wakeup Polarity<br>0b - Keep the LP_WAKEUP CFG0<br>1b - Invert the LP_WAKEUP CFG0 |
| **6**<br>**LP1_POL** | RW | LP_WAKEUP_CFG1 Wakeup Polarity<br>0b - Keep the LP_WAKEUP CFG1<br>1b - Invert the LP_WAKEUP CFG1 |
| **5**<br>**LP2_POL** | RW | LP_WAKEUP_CFG2 Wakeup Polarity<br>0b - Keep the LP_WAKEUP CFG2<br>1b - Invert the LP_WAKEUP CFG2 |
| **4**<br>**LP3_POL** | RW | LP_WAKEUP_CFG3 Wakeup Polarity<br>0b - Keep the LP_WAKEUP CFG3<br>1b - Invert the LP_WAKEUP CFG3 |
| **3**<br>**LP4_POL** | RW | LP_WAKEUP_CFG4 Wakeup Polarity<br>0b - Keep the LP_WAKEUP CFG4<br>1b - Invert the LP_WAKEUP CFG4 |
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Lock; can not be written |

Flagchip

# Chapter 25 System Control Module (SCM)

## 25.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • Changed the bit field range for "MAM0_S2_ECC" from "7-6" to "5-4" in the register section *SCM_MAM_ECCEN0;*<br>• Updated the description of the "DMA0_CFG_ECC" bit field in the register section *SCM_MAMECCEN1;*<br>• Updated the descriptions of the bit fields "GPR_SW_TRIG_7_4" and "GPR_SW_TRIG_3_0" in the register section *SCM_SOCMISC;*<br>• Removed the register section *SCM_FCSMU_SW;*<br>• Updated the bit field descriptions in the register section *SCM_SYSAP_CTRL;*<br>• Changed the name of the register "SCM_HSM_PCC (HSM_PCC Register)" to "SCM_SUBSYS_PCC (Subsystem PCC Register);<br>• Updated the bit field descriptions in the register section *SCM_SUBSYS_PCC;*<br>• Changed the name of the register "SCM_HSM_STATUS (HSM Status Register)" to "SCM_SUBSYS_STATUS (Subsystem Status Register);<br>• Changed the access attribute for the bit fields "DONE" and "ERR" from "RO" to "W1C" in the register section *SCM_CRCCSR;*<br>• In the register section *SCM_CRCRES:*<br>- Changed the access attribute from "RO" to "RW";<br>- Updated the description of the "RESULT" bit field.<br>• Fixed typos |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## 25.2 Chip-specific Information

N/A

## 25.3 Features

- Chip identification (ID) related information
- Chip configuration information about CPU lockstep, RAM size, and PFLASH size
- Support ECC monitor control on the master and slave
  - MAM ECC Enable Control
  - CPU ECC Enable Control

  ***Note:*** The value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.
- ECC enable control on CM7 ITCM/DTCM
- CM7 initial vectors (loaded from NVR)
- SRAM EDC error control
- SYSROM, HSMROM Power Gating Control
- Internal routing control on FCSPI/FCUART/ADC/FTU/FLEXCAN/MSC/ISM
- FTU miscellaneous configuration in chip level
- ADC miscellaneous configuration in chip level
- TRACE clock configuration
- Generation of 8 software triggers for the TRGSEL
- Provide sub-system low-power status information of all CM7 cores
- Matrix ECC Error recorder
- TPU GTBC Control
- Mailbox data input from debug to device
- Mailbox data output from device to debug
- HSM interface/control and status
- Master (DMA, HSM) Halt Bus Request
- NMI interrupt router
- Support the SCM Cyclic Redundancy Check (CRC) function
- Support Register Domain Write Permission
- Support Register Lock function

## 25.4 Overview

The main function of the System Control Module (SCM) is for the chip-level miscellaneous configuration. For all detailed functions, refer to corresponding chapters of different peripheral modules.

## 25.5 Functional Description

### 25.5.1 SCM CRC List

The SCM implements a CRC logic to monitor the SCM configuration registers.

| Type | CRC Type Register Name |
|---|---|
| Chip Configuration Information | CHIPCFG0, CHIPCFG1 |
| Matrix ECC Control | MAMECCEN0, MAMECCEN1, CPU0ECCEN |
| CPU Initial Vector | CPU0VTOR |
| Peripheral Routing | FCSPI_ROUTING, FCUART_ROUTING0, FCUART_ROUTING1, FTU_ROUTING, CAN_ROUTING, MSC0_ROUTING |
| Clock Divider | PERI_CLKDIV |

| Type | CRC Type Register Name |
|---|---|
| Interrupt Router | INT_ROUTER_NMI |

### 25.5.2 SCM Register LOCK Function

For the LOCK bit, the user needs first to configure other bit fields except the LOCK bit, then configure the LOCK bit if needed. When the LOCK bit is set, other bit fields cannot be written again, and only a power-on reset can clear the LOCK bit.

### 25.5.3 SCM Register WPB_LOCK and WPB Function

For WPB_LOCK and WPB bits, the user needs first to configure the WPB bit to assign the write permission to CPU0 (WPB = 0 & 1, CPU0 can write), then configure the WPB_LOCK bit if needed. When the WPB_LOCK bit is set, the corresponding WPB bit cannot be written again, but other bit fields can still be configured with the allowed CPU. And only the power-on reset can clear the WPB_LOCK bit.

## 25.6 Register Definition

SCM base address : 4007_2000h

**Register Overview - SCM**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | SCM_UIDL (Unique Identification Register 0) | 32 | RO | 0000_0000h | page |
| **4h** | SCM_UIDML (Unique Identification Register 1) | 32 | RO | 0000_0000h | page |
| **8h** | SCM_UIDMH (Unique Identification Register 2) | 32 | RO | 0000_0000h | page |
| **Ch** | SCM_UIDH (Unique Identification Register 3) | 32 | RO | 0000_0000h | page |
| **10h** | SCM_PARTID0 (PART ID Register) | 32 | RO | 0000_0300h | page |
| **18h** | SCM_CHIPCFG0 (CHIPCFG Register 0) | 32 | RO | 4100_0103h | page |
| **1Ch** | SCM_CHIPCFG1 (CHIPCFG Register 1) | 32 | RO | 00A8_02A0h | page |
| **20h** | SCM_MAMECCEN0 (MAM ECC Enable Register 0) | 32 | RW | 02AA_8A2Ah | page |
| **24h** | SCM_MAMECCEN1 (MAM ECC Enable Register 1) | 32 | RW | 0000_2922h | page |
| **28h** | SCM_CPU0ECCEN (CPU0 ECC Enable Register) | 32 | RW | 0000_0AAAh | page |
| **44h** | SCM_SRAM_EDC_CTRL (SRAM EDC Control Register) | 32 | RW | 0000_0000h | page |
| **4Ch** | SCM_ROMCFG (ROM Configuration Register) | 32 | RW | 0000_0000h | page |
| **50h** | SCM_FCSPI_ROUTING (FCSPI Routing Register) | 32 | RW | 0054_3210h | page |
| **54h** | SCM_FCUART_ROUTING0 (FCUART Routing Register 0) | 32 | RW | 7654_3210h | page |
| **60h** | SCM_ADC_ROUTING (ADC Routing Register) | 32 | RW | 0000_0000h | page |
| **64h** | SCM_FTU_ROUTING (FTU Routing Register) | 32 | RW | 0000_0000h | page |
| **68h** | SCM_FTU_GTBC (FTU Global Time Base Control Register) | 32 | RW | 0000_0000h | page |
| **6Ch** | SCM_FTU_SYNC (FTU SYNC Register) | 32 | RW | 0000_0000h | page |

| | | | | | |
|---|---|---|---|---|---|
| **70h** | SCM_DEBUG_TRACE (DEBUG TRACE Register) | 32 | RW | 0000_0000h | page |
| **74h** | SCM_SOCMISC (SOCMISC Register) | 32 | RW | 0000_0000h | page |
| **80h** | SCM_CCM0_STATUS (CCM0 Status Register) | 32 | RO | 0000_00D0h | page |
| **94h** | SCM_FLEXCAN_ROUTING (FLEXCAN Routing Register) | 32 | RW | 0000_0000h | page |
| **98h** | SCM_MSC0_ROUTING (MSC0 Routing Register) | 32 | RW | 0000_0000h | page |
| **A4h** | SCM_FCSMU_SW (FCSMU Software Trigger Register) | 32 | RW | 0000_0000h | page |
| **A8h** | SCM_ISM_ROUTING (ISM Routing Register) | 32 | RW | 0000_0000h | page |
| **B0h** | SCM_MATRIX_STATUS0 (Matrix Status Register 0) | 32 | W1C | 0000_0000h | page |
| **B4h** | SCM_MATRIX_STATUS1 (Matrix Status Register 1) | 32 | W1C | 0000_0000h | page |
| **B8h** | SCM_MATRIX_STATUS2 (Matrix Status Register 2) | 32 | W1C | 0000_0000h | page |
| **C4h** | SCM_MATRIX_STATUS5 (Matrix Status Register 5) | 32 | W1C | 0000_0000h | page |
| **C8h** | SCM_MATRIX_ID_STATUS0 (Matrix Master ID Status Register) | 32 | W1C | 0000_0000h | page |
| **CCh** | SCM_MATRIX_STATUS6 (Matrix Status Register 6) | 32 | W1C | 0000_0000h | page |
| **D0h** | SCM_MATRIX_STATUS7 (Matrix Status Register 7) | 32 | W1C | 0000_0000h | page |
| **F0h** | SCM_TPU_GTBCM (TPU Global Time Base Control Mask Register) | 32 | RW | 0000_0000h | page |
| **F4h** | SCM_FTU_GTBCM (FTU Global Time Base Control Mask Register) | 32 | RW | 0000_0000h | page |
| **100h** | SCM_SYSAP_MDO (SYSAP MDO Register) | 32 | RW | 0000_0000h | page |
| **104h** | SCM_SYSAP_MDI (SYSAP MDI Register) | 32 | RO | 0000_0000h | page |
| **108h** | SCM_SYSAP_CTRL (SYSAP Control Register) | 32 | RO | 0000_0000h | page |
| **110h** | SCM_SUBSYS_PCC (Subsystem PCC Register) | 32 | RW | 1080_0000h | page |
| **114h** | SCM_SUBSYS_STATUS (Subsystem Status Register) | 32 | RO | 0000_0080h | page |
| **118h** | SCM_MDO_FLAG (Mailbox Data Output Flag Register) | 32 | RO | 0000_0000h | page |
| **120h** | SCM_MASTER_HALT_REQ (Master Halt Request Register) | 32 | RW | 0000_0000h | page |
| **124h** | SCM_MASTER_HALT_ACK (Master Halt ACK Register) | 32 | RO | 0000_0000h | page |
| **1FCh** | SCM_INT_ROUTER_NMI (NMI Interrupt Router Register) | 32 | RW | 0100_0000h | page |
| **800h** | SCM_CRCCSR (CRC Control Status Register) | 32 | RW | 0000_0000h | page |
| **804h** | SCM_CRCRES (CRC Result Register) | 32 | RW | 0000_0000h | page |

### 25.6.1 SCM_UIDL (Unique Identification Register 0)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | UIDL | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | UIDL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_UIDL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 UIDL** | RO | Unique identification for the chip. Loaded from NVR. |

### 25.6.2 SCM_UIDML (Unique Identification Register 1)

Offset : 4h

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

**SCM_UIDML field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 UIDML** | RO | Unique identification for the chip. Loaded from NVR. |

### 25.6.3 SCM_UIDMH (Unique Identification Register 2)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | UDMH | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | UDMH | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_UIDMH field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 UDMH** | RO | Unique identification for the chip. Loaded from NVR. |

### 25.6.4 SCM_UIDH (Unique Identification Register 3)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | UIDH | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | UIDH | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_UIDH field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 UIDH** | RO | Unique identification for the chip. Loaded from NVR. |

### 25.6.5 SCM_PARTID0 (PART ID Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | | | | FAM_ID | | | | | | | | REVID | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_PARTID0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **11-4**<br>**FAM_ID** | RO | Family Identification |
| **3-0**<br>**REVID** | RO | Revision Identification |

### 25.6.6 SCM_CHIPCFG0 (CHIPCFG Register 0)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | CPU0_LOCKSTEP_EN | 0 | | | | | DMA0_LOCKSTEP_EN | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | | | | | | | CAN_FD | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**SCM_CHIPCFG0 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **30**<br>**CPU0_LOCKSTEP_EN** | RO | CPU0 LOCKSTEP ENABLE<br>0b - CPU0 lockstep disable<br>1b - CPU0 lockstep enable |
| **24**<br>**DMA0_LOCKSTEP_EN** | RO | DMA0 LOCKSTEP ENABLE<br>0b - DMA0 lockstep disable<br>1b - DMA0 lockstep enable |
| **8**<br>**CAN_FD** | RO | CANFD Enable<br>0b - CANFD disable<br>1b - CANFD enable |

### 25.6.7 SCM_CHIPCFG1 (CHIPCFG Register 1)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DEVICE_ID | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DEVICE_ID | | | | | | | | | | | | 0 | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

**SCM_CHIPCFG1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-4**<br>**DEVICE_ID** | RO | Device ID |

### 25.6.8 SCM_MAMECCEN0 (MAM ECC Enable Register 0)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WPB_LOCK | WPB | | | 0 | | SRAM1_ECC | | SRAM0_ECC | | MAM0_S8_ECC | | MAM0_S7_ECC | | MAM0_S6_ECC | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MAM0_S5_ECC | | 0 | | MAM0_S4_ECC | | MAM0_S3_ECC | | 0 | | MAM0_S2_ECC | | MAM0_S1_ECC | | MAM0_S0_ECC | |
| RST | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

**SCM_MAMECCEN0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**WPB_LOCK** | RW | Write Permission Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**WPB** | RW | Write Permission<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |
| **25-24**<br>**SRAM1_ECC** | RW | SRAM1 Array ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |
| **23-22**<br>**SRAM0_ECC** | RW | SRAM0 Array ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |
| **21-20**<br>**MAM0_S8_ECC** | RW | MAM0 S8 GPIO Slave ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |

| | | |
|---|---|---|
| **19-18**<br>**MAM0_S7_ECC** | RW | MAM0 S7 AFCB1 Slave ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |
| **17-16**<br>**MAM0_S6_ECC** | RW | MAM0 S6 AFCB0 Slave ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |
| **15-14**<br>**MAM0_S5_ECC** | RW | MAM0 S5 TCM Slave ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |
| **11-10**<br>**MAM0_S4_ECC** | RW | MAM0 S4 SRAM1 Slave ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |
| **9-8**<br>**MAM0_S3_ECC** | RW | MAM0 S3 SRAM0 Slave ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |

| | | |
|---|---|---|
| **5-4**<br>**MAM0_S2_ECC** | RW | MAM0 S2 DFlash Slave ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |
| **3-2**<br>**MAM0_S1_ECC** | RW | MAM0 S1 PFlash Slave ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |
| **1-0**<br>**MAM0_S0_ECC** | RW | MAM0 S0 ROM Array ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |

### 25.6.9 SCM_MAMECCEN1 (MAM ECC Enable Register 1)

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WPB_LOCK | WPB | | | | | | 0 | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | HSM_IRAM_ECC | | HSM_DRAM_ECC | | 0 | DMA0_CFG_ECC | 0 | | DMA0_ECC | | 0 | | HSM_ECC | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

**SCM_MAMECCEN1 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31**<br>**WPB_LOCK** | RW | Write Permission Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**WPB** | RW | Write Permission<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |
| **13-12**<br>**HSM_IRAM_ECC** | RW | HSM IRAM Memory ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |
| **11-10**<br>**HSM_DRAM_ECC** | RW | HSM DRAM Memory ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |
| **8**<br>**DMA0_CFG_ECC** | RW | DMA0 CFG Memory ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>0b - DMA0 CFG ECC Generate Enable During Write, ECC Check Disable During Read<br>1b - DMA0 CFG ECC Generate Enable During Write, ECC Check Enable During Read |
| **5-4**<br>**DMA0_ECC** | RW | DMA0 Master ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |

| | | |
|---|---|---|
| **1-0**<br>**HSM_ECC** | RW | HSM Master ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |

### 25.6.10 SCM_CPU0ECCEN (CPU0 ECC Enable Register)

Offset : 28h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WPB_LOCK | WPB | | | 0 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | CPU0_ITCM_ECC | | CPU0_DTCM1_ECC | | CPU0_DTCM0_ECC | | CPU0_AHBS_ECC | | CPU0_AHBP_ECC | | CPU0_AHBM_ECC | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

**SCM_CPU0ECCEN field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**WPB_LOCK** | RW | Write Permission Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**WPB** | RW | Write Permission<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |

| | | |
|---|---|---|
| **11-10**<br>**CPU0_ITCM_ECC** | RW | CPU0 ITCM ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |
| **9-8**<br>**CPU0_DTCM1_ECC** | RW | CPU0 DTCM1 ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |
| **7-6**<br>**CPU0_DTCM0_ECC** | RW | CPU0 DTCM0 ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |
| **5-4**<br>**CPU0_AHBS_ECC** | RW | CPU0 AHBS Master ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |
| **3-2**<br>**CPU0_AHBP_ECC** | RW | CPU0 AHBP Master ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |

| | | |
|---|---|---|
| **1-0**<br>**CPU0_AHBM_ECC** | RW | CPU0 AHBM Master ECC Enable Control<br>Note that the value for ECC ENABLE should be: NVR load value: 0x10; software program value: 0x11.<br>00b - Reserved<br>01b - Reserved<br>10b - ECC Generate Enable During Write, ECC Check Disable During Read<br>11b - ECC Generate Enable During Write, ECC Check Enable During Read |

### 25.6.11 SCM_SRAM_EDC_CTRL (SRAM EDC Control Register)

Offset : 44h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | SRAM1_EDC | SRAM0_EDC |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_SRAM_EDC_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **17**<br>**SRAM1_EDC** | RW | SRAM1 ECC Detect Control<br>0b - Disable EDC error when SRAM1 Multi Bit ECC Error Exist<br>1b - Only Enable EDC error when SRAM1 Multi Bit ECC Error Exist |
| **16**<br>**SRAM0_EDC** | RW | SRAM0 ECC Detect Control<br>0b - Disable EDC error when SRAM0 Multi Bit ECC Error Exist<br>1b - Only Enable EDC error when SRAM0 Multi Bit ECC Error Exist |

### 25.6.12 SCM_ROMCFG (ROM Configuration Register)

Offset : 4ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | HSMROM_PGEN | SYSROM_PGEN | 0 | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | LOCK |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_ROMCFG field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**HSMROM_PGEN** | RW | HSM ROM Power Gating Enable<br>0b - Not Power Gating<br>1b - Power Gating |
| **30**<br>**SYSROM_PGEN** | RW | System ROM Power Gating Enable<br>0b - Not Power Gating<br>1b - Power Gating |
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Locked; can not be written |

### 25.6.13 SCM_FCSPI_ROUTING (FCSPI Routing Register)

Offset : 50h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | SPI5_ROUTER | | | 0 | SPI4_ROUTER | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | SPI3_ROUTER | | | 0 | SPI2_ROUTER | | | 0 | SPI1_ROUTER | | | 0 | SPI0_ROUTER | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**SCM_FCSPI_ROUTING field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **22-20**<br>**SPI5_ROUTER** | RW | Use FCSPI5 to monitor other FCSPI's transmit interface.<br>000b - Monitor FCSPI0's transmit interface<br>001b - Monitor FCSPI1's transmit interface<br>010b - Monitor FCSPI2's transmit interface<br>011b - Monitor FCSPI3's transmit interface<br>100b - Monitor FCSPI4's transmit interface<br>101b - Monitor FCSPI5's transmit interface<br>110b - Reserved<br>111b - Reserved |
| **18-16**<br>**SPI4_ROUTER** | RW | Use FCSPI4 to monitor other FCSPI's transmit interface.<br>000b - Monitor FCSPI0's transmit interface<br>001b - Monitor FCSPI1's transmit interface<br>010b - Monitor FCSPI2's transmit interface<br>011b - Monitor FCSPI3's transmit interface<br>100b - Monitor FCSPI4's transmit interface<br>101b - Monitor FCSPI5's transmit interface<br>110b - Reserved<br>111b - Reserved |
| **14-12**<br>**SPI3_ROUTER** | RW | Use FCSPI3 to monitor other FCSPI's transmit interface.<br>000b - Monitor FCSPI0's transmit interface<br>001b - Monitor FCSPI1's transmit interface<br>010b - Monitor FCSPI2's transmit interface<br>011b - Monitor FCSPI3's transmit interface<br>100b - Monitor FCSPI4's transmit interface<br>101b - Monitor FCSPI5's transmit interface<br>110b - Reserved<br>111b - Reserved |
| **10-8**<br>**SPI2_ROUTER** | RW | Use FCSPI2 to monitor other FCSPI's transmit interface.<br>000b - Monitor FCSPI0's transmit interface<br>001b - Monitor FCSPI1's transmit interface<br>010b - Monitor FCSPI2's transmit interface<br>011b - Monitor FCSPI3's transmit interface<br>100b - Monitor FCSPI4's transmit interface<br>101b - Monitor FCSPI5's transmit interface<br>110b - Reserved<br>111b - Reserved |

| | | |
|---|---|---|
| **6-4**<br>**SPI1_ROUTER** | RW | Use FCSPI1 to monitor other FCSPI's transmit interface.<br>000b - Monitor FCSPI0's transmit interface<br>001b - Monitor FCSPI1's transmit interface<br>010b - Monitor FCSPI2's transmit interface<br>011b - Monitor FCSPI3's transmit interface<br>100b - Monitor FCSPI4's transmit interface<br>101b - Monitor FCSPI5's transmit interface<br>110b - Reserved<br>111b - Reserved |
| **2-0**<br>**SPI0_ROUTER** | RW | Use FCSPI0 to monitor other FCSPI's transmit interface.<br>000b - Monitor FCSPI0's transmit interface<br>001b - Monitor FCSPI1's transmit interface<br>010b - Monitor FCSPI2's transmit interface<br>011b - Monitor FCSPI3's transmit interface<br>100b - Monitor FCSPI4's transmit interface<br>101b - Monitor FCSPI5's transmit interface<br>110b - Reserved<br>111b - Reserved |

### 25.6.14 SCM_FCUART_ROUTING0 (FCUART Routing Register 0)

Offset : 54h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | UART7_ROUTER | | | | UART6_ROUTER | | | | UART5_ROUTER | | | | UART4_ROUTER | | | |
| RST | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | UART3_ROUTER | | | | UART2_ROUTER | | | | UART1_ROUTER | | | | UART0_ROUTER | | | |
| RST | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**SCM_FCUART_ROUTING0 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-28**<br>**UART7_ROUTER** | RW | Use FCUART7 to monitor other FCUART's transmit interface.<br>0000b - Monitor FCUART0's transmit interface<br>0001b - Monitor FCUART1's transmit interface<br>0010b - Monitor FCUART2's transmit interface<br>0011b - Monitor FCUART3's transmit interface<br>0100b - Monitor FCUART4's transmit interface<br>0101b - Monitor FCUART5's transmit interface<br>0110b - Monitor FCUART6's transmit interface<br>0111b - Monitor FCUART7's transmit interface<br>1000b - 1111b - Reserved |
| **27-24**<br>**UART6_ROUTER** | RW | Use FCUART6 to monitor other FCUART's transmit interface.<br>0000b - Monitor FCUART0's transmit interface<br>0001b - Monitor FCUART1's transmit interface<br>0010b - Monitor FCUART2's transmit interface<br>0011b - Monitor FCUART3's transmit interface<br>0100b - Monitor FCUART4's transmit interface<br>0101b - Monitor FCUART5's transmit interface<br>0110b - Monitor FCUART6's transmit interface<br>0111b - Monitor FCUART7's transmit interface<br>1000b - 1111b - Reserved |
| **23-20**<br>**UART5_ROUTER** | RW | Use FCUART5 to monitor other FCUART's transmit interface.<br>0000b - Monitor FCUART0's transmit interface<br>0001b - Monitor FCUART1's transmit interface<br>0010b - Monitor FCUART2's transmit interface<br>0011b - Monitor FCUART3's transmit interface<br>0100b - Monitor FCUART4's transmit interface<br>0101b - Monitor FCUART5's transmit interface<br>0110b - Monitor FCUART6's transmit interface<br>0111b - Monitor FCUART7's transmit interface<br>1000b - 1111b - Reserved |
| **19-16**<br>**UART4_ROUTER** | RW | Use FCUART4 to monitor other FCUART's transmit interface.<br>0000b - Monitor FCUART0's transmit interface<br>0001b - Monitor FCUART1's transmit interface<br>0010b - Monitor FCUART2's transmit interface<br>0011b - Monitor FCUART3's transmit interface<br>0100b - Monitor FCUART4's transmit interface<br>0101b - Monitor FCUART5's transmit interface<br>0110b - Monitor FCUART6's transmit interface<br>0111b - Monitor FCUART7's transmit interface<br>1000b - 1111b - Reserved |

| | | |
|---|---|---|
| **15-12**<br>**UART3_ROUTER** | RW | Use FCUART3 to monitor other FCUART's transmit interface.<br>0000b - Monitor FCUART0's transmit interface<br>0001b - Monitor FCUART1's transmit interface<br>0010b - Monitor FCUART2's transmit interface<br>0011b - Monitor FCUART3's transmit interface<br>0100b - Monitor FCUART4's transmit interface<br>0101b - Monitor FCUART5's transmit interface<br>0110b - Monitor FCUART6's transmit interface<br>0111b - Monitor FCUART7's transmit interface<br>1000b - 1111b - Reserved |
| **11-8**<br>**UART2_ROUTER** | RW | Use FCUART2 to monitor other FCUART's transmit interface.<br>0000b - Monitor FCUART0's transmit interface<br>0001b - Monitor FCUART1's transmit interface<br>0010b - Monitor FCUART2's transmit interface<br>0011b - Monitor FCUART3's transmit interface<br>0100b - Monitor FCUART4's transmit interface<br>0101b - Monitor FCUART5's transmit interface<br>0110b - Monitor FCUART6's transmit interface<br>0111b - Monitor FCUART7's transmit interface |
| **7-4**<br>**UART1_ROUTER** | RW | Use FCUART1 to monitor other FCUART's transmit interface.<br>0000b - Monitor FCUART0's transmit interface<br>0001b - Monitor FCUART1's transmit interface<br>0010b - Monitor FCUART2's transmit interface<br>0011b - Monitor FCUART3's transmit interface<br>0100b - Monitor FCUART4's transmit interface<br>0101b - Monitor FCUART5's transmit interface<br>0110b - Monitor FCUART6's transmit interface<br>0111b - Monitor FCUART7's transmit interface<br>1000b - 1111b - Reserved |
| **3-0**<br>**UART0_ROUTER** | RW | Use FCUART0 to monitor other FCUART's transmit interface.<br>0000b - Monitor FCUART0's transmit interface<br>0001b - Monitor FCUART1's transmit interface<br>0010b - Monitor FCUART2's transmit interface<br>0011b - Monitor FCUART3's transmit interface<br>0100b - Monitor FCUART4's transmit interface<br>0101b - Monitor FCUART5's transmit interface<br>0110b - Monitor FCUART6's transmit interface<br>0111b - Monitor FCUART7's transmit interface<br>1000b - 1111b - Reserved |

### 25.6.15 SCM_ADC_ROUTING (ADC Routing Register)

Offset : 60h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PTIMER01LOOP | ADC1_TRGSEL | ADC1_PRETRGSEL | | PTIMER01LOOP_1 | ADC1_SWPRETRG | | | PTIMER01LOOP_2 | ADC0_TRGSEL | ADC0_PRETRGSEL | | 0 | ADC0_SWPRETRG | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_ADC_ROUTING field descriptions**

| Field | Type | Description |
|---|---|---|
| **15**<br>**PTIMER01LOOP** | RW | PTIMER Loop Mode Select<br>Combined With PTIMER01LOOP_1 and PTIMER01LOOP_2 bit to use.<br>[PTIMER01LOOP, PTIMER01LOOP_1, PTIMER01LOOP_2]<br>000b - PTIMER0 CH0 loop, PTIMER1 CH0 loop<br>x01b - PTIMER1 CH0 and CH1 in loop mode<br>x10b - PTIMER0 CH0 and CH1 in loop mode<br>x11b - PTIMER1 CH0 and CH1 in loop mode, PTIMER0 CH0 and CH1 in loop mode<br>100b - PTIMER0 CH0 and PTIMER1 CH0 in loop mode |
| **14**<br>**ADC1_TRGSEL** | RW | ADC1 Trigger Source Select<br>Selects trigger source for ADC1.<br>0b - PTIMER output<br>1b - TRGSEL output |
| **13-12**<br>**ADC1_PRETRGSEL** | RW | ADC1 Pre-trigger Source Select<br>Selects pre-trigger source for ADC1.<br>00b - PTIMER pre-trigger<br>01b - Reserved<br>10b - SCM software pre-trigger<br>11b - Reserved |

| | | |
|---|---|---|
| **11**<br>**PTIMER01LOOP_1** | RW | PTIMER Loop Mode Select<br>Combined With PTIMER01LOOP_1 and PTIMER01LOOP_2 bit to use.[PTIMER01LOOP, PTIMER01LOOP_1, PTIMER01LOOP_2]<br>000b - PTIMER0 CH0 loop, PTIMER1 CH0 loop<br>x01b - PTIMER1 CH0 and CH1 in loop mode<br>x10b - PTIMER0 CH0 and CH1 in loop mode<br>x11b - PTIMER1 CH0 and CH1 in loop mode, PTIMER0 CH0 and CH1 in loop mode<br>100b - PTIMER0 CH0 and PTIMER1 CH0 in loop mode |
| **10-8**<br>**ADC1_SWPRETRG** | RW | ADC1 SCM Pre-trigger Sources<br>000b - SCM Software pre-trigger disabled<br>100b - SCM Software pre-trigger 0<br>101b - SCM Software pre-trigger 1<br>110b - SCM Software pre-trigger 2<br>111b - SCM Software pre-trigger 3 |
| **7**<br>**PTIMER01LOOP_2** | RW | PTIMER Loop Mode Select<br>Combined With PTIMER01LOOP_1 and PTIMER01LOOP_2 bit to use.[PTIMER01LOOP, PTIMER01LOOP_1, PTIMER01LOOP_2]<br>000b - PTIMER0 CH0 loop, PTIMER1 CH0 loop<br>x01b - PTIMER1 CH0 and CH1 in loop mode<br>x10b - PTIMER0 CH0 and CH1 in loop mode<br>x11b - PTIMER1 CH0 and CH1 in loop mode, PTIMER0 CH0 and CH1 in loop mode<br>100b - PTIMER0 CH0 and PTIMER1 CH0 in loop mode |
| **6**<br>**ADC0_TRGSEL** | RW | ADC0 Trigger Source Select<br>Selects trigger source for ADC0.<br>0b - PTIMER output<br>1b - TRGSEL output |
| **5-4**<br>**ADC0_PRETRGSEL** | RW | ADC0 Pre-trigger Source Select<br>Selects pre-trigger source for ADC0.<br>00b - PTIMER pre-trigger<br>01b - Reserved<br>10b - SCM software pre-trigger<br>11b - Reserved |
| **2-0**<br>**ADC0_SWPRETRG** | RW | ADC0 Software Pre-trigger Sources<br>000b - SCM Software pre-trigger disabled<br>100b - SCM Software pre-trigger 0<br>101b - SCM Software pre-trigger 1<br>110b - SCM Software pre-trigger 2<br>111b - SCM Software pre-trigger 3 |

### 25.6.16 SCM_FTU_ROUTING (FTU Routing Register)

Offset : 64h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FTU3_OUTSEL | | | | | | | | FTU0_OUTSEL | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | FTU3_CH0SEL | 0 | | | FTU2_CH1SEL | FTU2_CH0SEL | | 0 | | FTU1_CH0SEL | | 0 | | | LOCK |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_FTU_ROUTING field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**FTU3_OUTSEL** | RW | FTU3_OUTSEL<br>Select the FTU3 output source, should be ignored when FTU3 in input mode<br>0b - FTU3 CHx output will be directly routed to corresponding pad<br>1b - FTU3 pad output is equal to FTU3 CHx output & FTU2 CH1 output |
| **23-16**<br>**FTU0_OUTSEL** | RW | FTU0_OUTSEL<br>Select the FTU0 output source, should be ignored when FTU0 in input mode<br>0b - FTU0 CHx output will be directly routed to corresponding pad<br>1b - FTU0 pad output is equal to FTU0 CHx output & FTU1 CH1 output |
| **14**<br>**FTU3_CH0SEL** | RW | FTU3 CH0 Select<br>Selects FTU3 CH0 input<br>0b - FTU3_CH0 input<br>1b - XOR of FTU3_CH0,FTU3_CH1,and FTU3_CH2 |
| **10**<br>**FTU2_CH1SEL** | RW | FTU2 CH1 Select<br>Selects FTU2 CH1 input<br>0b - FTU2_CH1 input<br>1b - XOR of FTU2_CH0,FTU2_CH1,and FTU2_CH2 |

| | | |
|---|---|---|
| **9-8**<br>**FTU2_CH0SEL** | RW | FTU2 CH0 Select<br>Selects FTU2 CH0 input<br>00b - FTU2_CH0 input<br>01b - CMP0 output<br>10b - CMP1 output<br>11b - Reserved |
| **5-4**<br>**FTU1_CH0SEL** | RW | FTU1 CH0 Select<br>Selects FTU1 CH0 input<br>00b - FTU1_CH0 input<br>01b - CMP0 output<br>10b - CMP1 output<br>11b - Reserved |
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Lock; can not be written |

### 25.6.17 SCM_FTU_GTBC (FTU Global Time Base Control Register)

Offset : 68h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | FTU7_GTBC | FTU6_GTBC | FTU5_GTBC | FTU4_GTBC | FTU3_GTBC | FTU2_GTBC | FTU1_GTBC | FTU0_GTBC |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | TPU_GTBC | 0 | | | | | | | | | | | LOCK |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_FTU_GTBC field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**FTU7_GTBC** | RW | FTU7 Global Time Base Control Select<br>0b - FTU7 do not updated.<br>1b - FTU7 updated |
| **22**<br>**FTU6_GTBC** | RW | FTU6 Global Time Base Control Select<br>0b - FTU6 do not updated.<br>1b - FTU6 updated |

| | | |
|---|---|---|
| **21**<br>**FTU5_GTBC** | RW | FTU5 Global Time Base Control Select<br>0b - FTU5 do not updated.<br>1b - FTU5 updated |
| **20**<br>**FTU4_GTBC** | RW | FTU4 Global Time Base Control Select<br>0b - FTU4 do not updated.<br>1b - FTU4 updated |
| **19**<br>**FTU3_GTBC** | RW | FTU3 Global Time Base Control Select<br>0b - FTU3 do not updated.<br>1b - FTU3 updated |
| **18**<br>**FTU2_GTBC** | RW | FTU2 Global Time Base Control Select<br>0b - FTU2 do not updated.<br>1b - FTU2 updated |
| **17**<br>**FTU1_GTBC** | RW | FTU1 Global Time Base Control Select<br>0b - FTU1 do not updated.<br>1b - FTU1 updated |
| **16**<br>**FTU0_GTBC** | RW | FTU0 Global Time Base Control Select<br>0b - FTU0 do not updated.<br>1b - FTU0 updated |
| **12**<br>**TPU_GTBC** | RW | TPU Global Time Base Control<br>0b - ETPU Global Time Base Disable<br>1b - ETPU Global Time Base Enable |
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Lock; can not be written |

### 25.6.18 SCM_FTU_SYNC (FTU SYNC Register)

Offset : 6Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | FTU7 | 0 | FTU6 | 0 | FTU5 | 0 | FTU4 | 0 | FTU3 | 0 | FTU2 | 0 | FTU1 | 0 | FTU0 |
| W | | SYNC | | SYNC | | SYNC | | SYNC | | SYNC | | SYNC | | SYNC | | SYNC |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_FTU_SYNC field descriptions**

| Field | Type | Description |
|---|---|---|
| **14**<br>**FTU7SYNC** | RW | FTU7 Sync Control<br>This is used as trigger source for FTU7. For details, please see section FTU Hardware Triggers and Synchronization. |
| **12**<br>**FTU6SYNC** | RW | FTU6 Sync Control<br>This is used as trigger source for FTU6. For details, please see section FTU Hardware Triggers and Synchronization. |
| **10**<br>**FTU5SYNC** | RW | FTU5 Sync Control<br>This is used as trigger source for FTU5. For details, please see section FTU Hardware Triggers and Synchronization. |
| **8**<br>**FTU4SYNC** | RW | FTU4 Sync Control<br>This is used as trigger source for FTU4. For details, please see section FTU Hardware Triggers and Synchronization. |
| **6**<br>**FTU3SYNC** | RW | FTU3 Sync Control<br>This is used as trigger source for FTU3. For details, please see section FTU Hardware Triggers and Synchronization. |
| **4**<br>**FTU2SYNC** | RW | FTU2 Sync Control<br>This is used as trigger source for FTU2. For details, please see section FTU Hardware Triggers and Synchronization. |
| **2**<br>**FTU1SYNC** | RW | FTU1 Sync Control<br>This is used as trigger source for FTU1. For details, please see section FTU Hardware Triggers and Synchronization. |
| **0**<br>**FTU0SYNC** | RW | FTU0 Sync Control<br>This is used as trigger source for FTU0. For details, please see section FTU Hardware Triggers and Synchronization. |

### 25.6.19 SCM_DEBUG_TRACE (DEBUG TRACE Register)

Offset : 70h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DEBUG_ATCLK_EN | 0 | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TRACECLK_DIV | | | | TRAC | TRAC | 0 | LOCK |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | ECLK_SEL | ECLK_EN | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_DEBUG_TRACE field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**DEBUG_ATCLK_EN** | RW | Debug ATCLK Enable |
| **7-4**<br>**TRACECLK_DIV** | RW | Trace Clock Divider Value |
| **3**<br>**TRACECLK_SEL** | RW | Trace Clock Select<br>0b - Select the core clock as the trace clock<br>1b - Select SCG PLL1 DIVH as the trace clock |
| **2**<br>**TRACECLK_EN** | RW | Trace Clock Divider Enable |
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Lock; can not be written |

### 25.6.20 SCM_SOCMISC (SOCMISC Register)

Offset : 74h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WPB_LOCK | WPB | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | GPR_SW_TRIG_7_4 | | | | GPR_SW_TRIG_3_0 | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_SOCMISC field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31**<br>**WPB_LOCK** | RW | Write Permission Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**WPB** | RW | Write Permission<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |
| **7-4**<br>**GPR_SW_TRIG_7_4** | RW | Software Trigger 4~7 to TRGSEL1&TRGSEL5<br>0001b - Generate software trigger4 to TRGSEL1&TRGSEL5 source 2<br>0010b - Generate software trigger5 to TRGSEL1&TRGSEL5 source 3<br>0100b - Generate software trigger6 to TRGSEL1&TRGSEL5 source 4<br>1000b - Generate software trigger7 to TRGSEL1&TRGSEL5 source 5 |
| **3-0**<br>**GPR_SW_TRIG_3_0** | RW | Software Trigger 0~3 to TRGSEL0&TRGSEL4<br>0001b - Generate software trigger0 to TRGSEL0&TRGSEL4 source 2<br>0010b - Generate software trigger1 to TRGSEL0&TRGSEL4 source 3<br>0100b - Generate software trigger2 to TRGSEL0&TRGSEL4 source 4<br>1000b - Generate software trigger3 to TRGSEL0&TRGSEL4 source 5 |

### 25.6.21 SCM_CCM0_STATUS (CCM0 Status Register)

Offset : 80h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CPU0_STOP_SYS_SLAVE | CPU0_STOP_MASTER | CPU0_STOP_SLOW_SLAVE | CPU0_STOP_BUS_SLAVE | 0 | | CPU0_DEEPSLEEPING | CPU0_SLEEPING |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

**SCM_CCM0_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **7**<br>**CPU0_STOP_SYS_SLAVE** | RO | CCM0 FCG Stop Sys Slave Ack |
| **6**<br>**CPU0_STOP_MASTER** | RO | CCM0 FCG Stop Master Ack |
| **5**<br>**CPU0_STOP_SLOW_SLAVE** | RO | CCM0 FCG Stop Slow Slave Ack |
| **4**<br>**CPU0_STOP_BUS_SLAVE** | RO | CCM0 FCG Stop Bus Slave Ack |
| **1**<br>**CPU0_DEEPSLEEPING** | RO | CPU0 Sleepdeep |
| **0**<br>**CPU0_SLEEPING** | RO | CPU0 Sleeping |

### 25.6.22 SCM_FLEXCAN_ROUTING (FLEXCAN Routing Register)

Offset : 94h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FLEXCAN_TRIGGER_MASK | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | LOCK |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_FLEXCAN_ROUTING field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-16**<br>**FLEXCAN_TRIGGER_MASK** | RW | Select TRESEL input from pad TRGSEL_IN0~TRGSEL_IN15 for FLEXCAN0/1/2/3 RX<br>0b - Disable Select TRESEL input from pad TRGSEL_IN0~TRGSEL_IN15 for FLEXCAN0/1/2/3 RX<br>1b - Enable Select TRESEL input from pad TRGSEL_IN0~TRGSEL_IN15 for FLEXCAN0/1/2/3 RX |

| | | |
|---|---|---|
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Lock; can not be written |

### 25.6.23 SCM_MSC0_ROUTING (MSC0 Routing Register)

Offset : 98h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MSC0_31_24_SEL | | | | MSC0_23_16_SEL | | | | MSC0_15_8_SEL | | | | MSC0_7_0_SEL | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | LOCK |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_MSC0_ROUTING field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-28**<br>**MSC0_31_24_SEL** | RW | MSC0 ALTIN[31:24] Source Select<br>0000b - Select FTU0 ch do out[7:0] to MSC0 ALTIN[31:24]<br>0001b - Select FTU1 ch do out[7:0] to MSC0 ALTIN[31:24]<br>0010b - Select FTU2 ch do out[7:0] to MSC0 ALTIN[31:24]<br>0011b - Select FTU3 ch do out[7:0] to MSC0 ALTIN[31:24]<br>0100b - Select FTU4 ch do out[7:0] to MSC0 ALTIN[31:24]<br>0101b - Select FTU5 ch do out[7:0] to MSC0 ALTIN[31:24]<br>0110b - Select FTU6 ch do out[7:0] to MSC0 ALTIN[31:24]<br>0111b - Select FTU7 ch do out[7:0] to MSC0 ALTIN[31:24]<br>1000b - Reserved<br>1001b - Reserved<br>1010b - Reserved<br>1011b - Reserved<br>1100b - Reserved<br>1101b - Reserved<br>1110b - Reserved<br>1111b - Reserved |

| | | |
|---|---|---|
| **27-24**<br>**MSC0_23_16_SEL** | RW | MSC0 ALTIN[23:16] Source Select<br>0000b - Select FTU0 ch do out[7:0] to MSC0 ALTIN[23:16]<br>0001b - Select FTU1 ch do out[7:0] to MSC0 ALTIN[23:16]<br>0010b - Select FTU2 ch do out[7:0] to MSC0 ALTIN[23:16]<br>0011b - Select FTU3 ch do out[7:0] to MSC0 ALTIN[23:16]<br>0100b - Select FTU4 ch do out[7:0] to MSC0 ALTIN[23:16]<br>0101b - Select FTU5 ch do out[7:0] to MSC0 ALTIN[23:16]<br>0110b - Select FTU6 ch do out[7:0] to MSC0 ALTIN[23:16]<br>0111b - Select FTU7 ch do out[7:0] to MSC0 ALTIN[23:16]<br>1000b - Reserved<br>1001b - Reserved<br>1010b - Reserved<br>1011b - Reserved<br>1100b - Reserved<br>1101b - Reserved<br>1110b - Reserved<br>1111b - Reserved |
| **23-20**<br>**MSC0_15_8_SEL** | RW | MSC0 ALTIN[15:8] Source Select<br>0000b - Select FTU0 ch do out[7:0] to MSC0 ALTIN[15:8]<br>0001b - Select FTU1 ch do out[7:0] to MSC0 ALTIN[15:8]<br>0010b - Select FTU2 ch do out[7:0] to MSC0 ALTIN[15:8]<br>0011b - Select FTU3 ch do out[7:0] to MSC0 ALTIN[15:8]<br>0100b - Select FTU4 ch do out[7:0] to MSC0 ALTIN[15:8]<br>0101b - Select FTU5 ch do out[7:0] to MSC0 ALTIN[15:8]<br>0110b - Select FTU6 ch do out[7:0] to MSC0 ALTIN[15:8]<br>0111b - Select FTU7 ch do out[7:0] to MSC0 ALTIN[15:8]<br>1000b - Reserved<br>1001b - Reserved<br>1010b - Reserved<br>1011b - Reserved<br>1100b - Reserved<br>1101b - Reserved<br>1110b - Reserved<br>1111b - Reserved |

| | | |
|---|---|---|
| **19-16**<br>**MSC0_7_0_SEL** | RW | MSC0 ALTIN[7:0] Source Select<br>0000b - Select FTU0 ch do out[7:0] to MSC0 ALTIN[7:0]<br>0001b - Select FTU1 ch do out[7:0] to MSC0 ALTIN[7:0]<br>0010b - Select FTU2 ch do out[7:0] to MSC0 ALTIN[7:0]<br>0011b - Select FTU3 ch do out[7:0] to MSC0 ALTIN[7:0]<br>0100b - Select FTU4 ch do out[7:0] to MSC0 ALTIN[7:0]<br>0101b - Select FTU5 ch do out[7:0] to MSC0 ALTIN[7:0]<br>0110b - Select FTU6 ch do out[7:0] to MSC0 ALTIN[7:0]<br>0111b - Select FTU7 ch do out[7:0] to MSC0 ALTIN[7:0]<br>1000b - Reserved<br>1001b - Reserved<br>1010b - Reserved<br>1011b - Reserved<br>1100b - Reserved<br>1101b - Reserved<br>1110b - Reserved<br>1111b - Reserved |
| **0**<br>**LOCK** | RW | 0b - Unlock; can be written<br>1b - Lock; can not be written |

### 25.6.24 SCM_ISM_ROUTING (ISM Routing Register)

Offset : A8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ISM_ROUT_FTU_D | | | | ISM_ROUT_FTU_C | | | | ISM_ROUT_FTU_B | | | | ISM_ROUT_FTU_A | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_ISM_ROUTING field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **15-12**<br>**ISM_ROUT_FTU_D** | RW | FTU_IN[15:8] Source Select<br>0000b - Select FTU0 ch fcf ind[7:0] to TRGSEL3 source [37:30]<br>0001b - Select FTU1 ch fcf ind[7:0] to TRGSEL3 source [37:30]<br>0010b - Select FTU2 ch fcf ind[7:0] to TRGSEL3 source [37:30]<br>0011b - Select FTU3 ch fcf ind[7:0] to TRGSEL3 source [37:30]<br>0100b - Select FTU4 ch fcf ind[7:0] to TRGSEL3 source [37:30]<br>0101b - Select FTU5 ch fcf ind[7:0] to TRGSEL3 source [37:30]<br>0110b - Select FTU6 ch fcf ind[7:0] to TRGSEL3 source [37:30]<br>0111b - Select FTU7 ch fcf ind[7:0] to TRGSEL3 source [37:30]<br>1000b - Reserved<br>1001b - Reserved<br>1010b - Reserved<br>1011b - Reserved<br>1100b - Reserved<br>1101b - Reserved<br>1110b - Reserved<br>1111b - Reserved |
| **11-8**<br>**ISM_ROUT_FTU_C** | RW | FTU_IN[7:0] Source Select<br>0000b - Select FTU0 ch fcf ind[7:0] to TRGSEL3 source [29:22]<br>0001b - Select FTU1 ch fcf ind[7:0] to TRGSEL3 source [29:22]<br>0010b - Select FTU2 ch fcf ind[7:0] to TRGSEL3 source [29:22]<br>0011b - Select FTU3 ch fcf ind[7:0] to TRGSEL3 source [29:22]<br>0100b - Select FTU4 ch fcf ind[7:0] to TRGSEL3 source [29:22]<br>0101b - Select FTU5 ch fcf ind[7:0] to TRGSEL3 source [29:22]<br>0110b - Select FTU6 ch fcf ind[7:0] to TRGSEL3 source [29:22]<br>0111b - Select FTU7 ch fcf ind[7:0] to TRGSEL3 source [29:22]<br>1000b - Reserved<br>1001b - Reserved<br>1010b - Reserved<br>1011b - Reserved<br>1100b - Reserved<br>1101b - Reserved<br>1110b - Reserved<br>1111b - Reserved |

| | | |
|---|---|---|
| **7-4**<br>**ISM_ROUT_FTU_B** | RW | FTU_OUT[15:8] Source Select<br>0000b - Select FTU0 ch do out[7:0] to TRGSEL3 source [53:46]<br>0001b - Select FTU1 ch do out[7:0] to TRGSEL3 source [53:46]<br>0010b - Select FTU2 ch do out[7:0] to TRGSEL3 source [53:46]<br>0011b - Select FTU3 ch do out[7:0] to TRGSEL3 source [53:46]<br>0100b - Select FTU4 ch do out[7:0] to TRGSEL3 source [53:46]<br>0101b - Select FTU5 ch do out[7:0] to TRGSEL3 source [53:46]<br>0110b - Select FTU6 ch do out[7:0] to TRGSEL3 source [53:46]<br>0111b - Select FTU7 ch do out[7:0] to TRGSEL3 source [53:46]<br>1000b - Reserved<br>1001b - Reserved<br>1010b - Reserved<br>1011b - Reserved<br>1100b - Reserved<br>1101b - Reserved<br>1110b - Reserved<br>1111b - Reserved |
| **3-0**<br>**ISM_ROUT_FTU_A** | RW | FTU_OUT[7:0] Source Select<br>0000b - Select FTU0 ch do out[7:0] to TRGSEL3 source [45:38]<br>0001b - Select FTU1 ch do out[7:0] to TRGSEL3 source [45:38]<br>0010b - Select FTU2 ch do out[7:0] to TRGSEL3 source [45:38]<br>0011b - Select FTU3 ch do out[7:0] to TRGSEL3 source [45:38]<br>0100b - Select FTU4 ch do out[7:0] to TRGSEL3 source [45:38]<br>0101b - Select FTU5 ch do out[7:0] to TRGSEL3 source [45:38]<br>0110b - Select FTU6 ch do out[7:0] to TRGSEL3 source [45:38]<br>0111b - Select FTU7 ch do out[7:0] to TRGSEL3 source [45:38]<br>1000b - Reserved<br>1001b - Reserved<br>1010b - Reserved<br>1011b - Reserved<br>1100b - Reserved<br>1101b - Reserved<br>1110b - Reserved<br>1111b - Reserved |

### 25.6.25 SCM_MATRIX_STATUS0 (Matrix Status Register 0)

The register is POR Only.

0b - Error not occurred

1b - Error exists; write one to clear

Offset : B0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAM0_S5_M | MAM0_S5_AT | MAM0_S5_D | MAM0_S5_A | 0 | | | | MAM0_S4_M | MAM0_S4_AT | MAM0_S4_D | MAM0_S4_A | MAM0_S3_M | MAM0_S3_AT | MAM0_S3_D | MAM0_S3_A |
| W | w1c | w1c | w1c | w1c | | | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAM0_S2_M | MAM0_S2_AT | MAM0_S2_D | MAM0_S2_A | 0 | | | | MAM0_S1_M | MAM0_S1_AT | MAM0_S1_D | MAM0_S1_A | ROM_M | 0 | ROM_D | ROM_A |
| W | w1c | w1c | w1c | w1c | | | | | w1c | w1c | w1c | w1c | w1c | | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_MATRIX_STATUS0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31 MAM0_S5_M** | W1C | MAM0_S5 Slave ECC Multi Bit Data Error Flag |
| **30 MAM0_S5_AT** | W1C | MAM0_S5 Slave ECC Attribute Error Flag |
| **29 MAM0_S5_D** | W1C | MAM0_S5 Slave ECC Multi Bit Data Error Flag |
| **28 MAM0_S5_A** | W1C | MAM0_S5 Slave ECC Address Error Flag |
| **23 MAM0_S4_M** | W1C | MAM0_S4 Slave ECC Multi Bit Data Error Flag |
| **22 MAM0_S4_AT** | W1C | MAM0_S4 Slave ECC Attribute Error Flag |
| **21 MAM0_S4_D** | W1C | MAM0_S4 Slave ECC Multi Bit Data Error Flag |
| **20 MAM0_S4_A** | W1C | MAM0_S4 Slave ECC Address Error Flag |
| **19 MAM0_S3_M** | W1C | MAM0_S3 Slave ECC Multi Bit Data Error Flag |

| | | |
|---|---|---|
| **18**<br>**MAM0_S3_AT** | W1C | MAM0_S3 Slave ECC Attribute Error Flag |
| **17**<br>**MAM0_S3_D** | W1C | MAM0_S3 Slave ECC Multi Bit Data Error Flag |
| **16**<br>**MAM0_S3_A** | W1C | MAM0_S3 Slave ECC Address Error Flag |
| **15**<br>**MAM0_S2_M** | W1C | MAM0_S2 Slave ECC Multi Bit Data Error Flag |
| **14**<br>**MAM0_S2_AT** | W1C | MAM0_S2 Slave ECC Attribute Error Flag |
| **13**<br>**MAM0_S2_D** | W1C | MAM0_S2 Slave ECC Multi Bit Data Error Flag |
| **12**<br>**MAM0_S2_A** | W1C | MAM0_S2 Slave ECC Address Error Flag |
| **7**<br>**MAM0_S1_M** | W1C | MAM0_S1 Slave ECC Multi Bit Data Error Flag |
| **6**<br>**MAM0_S1_AT** | W1C | MAM0_S1 Slave ECC Attribute Error Flag |
| **5**<br>**MAM0_S1_D** | W1C | MAM0_S1 Slave ECC Multi Bit Data Error Flag |
| **4**<br>**MAM0_S1_A** | W1C | MAM0_S1 Slave ECC Address Error Flag |
| **3**<br>**ROM_M** | W1C | ROM Array ECC Multi Bit Data Error Flag |
| **1**<br>**ROM_D** | W1C | ROM Array ECC Single Bit Data Error Flag |
| **0**<br>**ROM_A** | W1C | ROM Array ECC Address Error Flag |

### 25.6.26 SCM_MATRIX_STATUS1 (Matrix Status Register 1)

The register is POR Only.

0b - Error not occurred

1b - Error exists; write one to clear

Offset : B4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DMA0_M | 0 | DMA0_D | DMA0_A | HSM_M | 0 | HSM_D | HSM_A | 0 | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | w1c | | w1c | w1c | w1c | | w1c | w1c | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAM0_S5_DS_M | 0 | MAM0_S5_DS_D | MAM0_S5_DS_A | MAM0_S8_M | MAM0_S8_AT | MAM0_S8_D | MAM0_S8_A | MAM0_S7_M | MAM0_S7_AT | MAM0_S7_D | MAM0_S7_A | MAM0_S6_M | MAM0_S6_AT | MAM0_S6_D | MAM0_S6_A |
| W | w1c | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_MATRIX_STATUS1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **27**<br>**DMA0_M** | W1C | DMA0 Master ECC Multi Bit Data Error Flag |
| **25**<br>**DMA0_D** | W1C | DMA0 Master ECC Single Bit Data Error Flag |
| **24**<br>**DMA0_A** | W1C | DMA0 Master ECC Address Error Flag |
| **23**<br>**HSM_M** | W1C | HSM Master ECC Multi Bit Data Error Flag |
| **21**<br>**HSM_D** | W1C | HSM Master ECC Single Bit Data Error Flag |
| **20**<br>**HSM_A** | W1C | HSM Master ECC Address Error Flag |
| **15**<br>**MAM0_S5_DS_M** | W1C | MAM0_S5 Downsize ECC Multi Bit Data Error Flag |
| **13**<br>**MAM0_S5_DS_D** | W1C | MAM0_S5 Downsize ECC Single Bit Data Error Flag |
| **12**<br>**MAM0_S5_DS_A** | W1C | MAM0_S5 Downsize ECC Address Error Flag |
| **11**<br>**MAM0_S8_M** | W1C | MAM0_S8 Slave ECC Multi Bit Data Error Flag |

| | | |
|---|---|---|
| **10**<br>**MAM0_S8_AT** | W1C | MAM0_S8 Slave ECC Attribute Error Flag |
| **9**<br>**MAM0_S8_D** | W1C | MAM0_S8 Slave ECC Multi Bit Data Error Flag |
| **8**<br>**MAM0_S8_A** | W1C | MAM0_S8 Slave ECC Address Error Flag |
| **7**<br>**MAM0_S7_M** | W1C | MAM0_S7 Slave ECC Multi Bit Data Error Flag |
| **6**<br>**MAM0_S7_AT** | W1C | MAM0_S7 Slave ECC Attribute Error Flag |
| **5**<br>**MAM0_S7_D** | W1C | MAM0_S7 Slave ECC Multi Bit Data Error Flag |
| **4**<br>**MAM0_S7_A** | W1C | MAM0_S7 Slave ECC Address Error Flag |
| **3**<br>**MAM0_S6_M** | W1C | MAM0_S6 Slave ECC Multi Bit Data Error Flag |
| **2**<br>**MAM0_S6_AT** | W1C | MAM0_S6 Slave ECC Attribute Error Flag |
| **1**<br>**MAM0_S6_D** | W1C | MAM0_S6 Slave ECC Single Bit Data Error Flag |
| **0**<br>**MAM0_S6_A** | W1C | MAM0_S6 Slave ECC Address Error Flag |

### 25.6.27 SCM_MATRIX_STATUS2 (Matrix Status Register 2)

The register is POR Only.

0b - Error not occurred

1b - Error exists; write one to clear

Offset : B8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | C0_OVERLAY | 0 | C0_LOCKSTEP | C0_DTCM1_A | C0_DTCM1_M | C0_DTCM1_S | C0_DTCM0_A | C0_DTCM0_M | C0_DTCM0_S | C0_ITCM_A | C0_ITCM_M | C0_ITCM_S | C0_DCACHE | C0_ICACHE | C0_DTCM1_DEC | C0_DTCM0_DEC |
| W | w1c | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | C0_ITCM_DEC | 0 | | C0_AHBP_F2S | C0_AHBM_M | 0 | C0_AHBM_D | C0_AHBM_A | C0_AHBS_M | 0 | C0_AHBS_D | C0_AHBS_A | C0_AHBP_M | 0 | C0_AHBP_D | C0_AHBP_A |
| W | w1c | | | w1c | w1c | | w1c | w1c | w1c | | w1c | w1c | w1c | | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_MATRIX_STATUS2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31<br>C0_OVERLAY** | W1C | CPU0 AHB Overlay Error Flag |
| **29<br>C0_LOCKSTEP** | W1C | CPU0 Lockstep Error Flag |
| **28<br>C0_DTCM1_A** | W1C | CPU0 DTCM1 ECC Address Error Flag |
| **27<br>C0_DTCM1_M** | W1C | CPU0 DTCM1 ECC Multi Bit Data Error Flag |
| **26<br>C0_DTCM1_S** | W1C | CPU0 DTCM1 ECC Single Bit Data Error Flag |
| **25<br>C0_DTCM0_A** | W1C | CPU0 DTCM0 ECC Address Error Flag |
| **24<br>C0_DTCM0_M** | W1C | CPU0 DTCM0 ECC Multi Bit Data Error Flag |
| **23<br>C0_DTCM0_S** | W1C | CPU0 DTCM0 ECC Single Bit Data Error Flag |
| **22<br>C0_ITCM_A** | W1C | CPU0 ITCM ECC Address Error Flag |
| **21<br>C0_ITCM_M** | W1C | CPU0 ITCM ECC Multi Bit Data Error Flag |
| **20<br>C0_ITCM_S** | W1C | CPU0 ITCM ECC Single Bit Data Error Flag |

| | | |
|---|---|---|
| **19**<br>**C0_DCACHE** | W1C | CPU0 DCache Array Error Flag |
| **18**<br>**C0_ICACHE** | W1C | CPU0 ICache Array Error Flag |
| **17**<br>**C0_DTCM1_DEC** | W1C | CPU0 DTCM1 Decode Monitor Error Flag |
| **16**<br>**C0_DTCM0_DEC** | W1C | CPU0 DTCM0 Decode Monitor Error Flag |
| **15**<br>**C0_ITCM_DEC** | W1C | CPU0 ITCM Decode Monitor Error Flag |
| **12**<br>**C0_AHBP_F2S** | W1C | CPU0 AHBP Fast to Slow Monitor Error Flag |
| **11**<br>**C0_AHBM_M** | W1C | CPU0 AHBM Master ECC Multi Bit Data Error Flag |
| **9**<br>**C0_AHBM_D** | W1C | CPU0 AHBM Master ECC Single Bit Data Error Flag |
| **8**<br>**C0_AHBM_A** | W1C | CPU0 AHBM Master ECC Address Error Flag |
| **7**<br>**C0_AHBS_M** | W1C | CPU0 AHBS Master ECC Multi Bit Data Error Flag |
| **5**<br>**C0_AHBS_D** | W1C | CPU0 AHBS Master ECC Single Bit Data Error Flag |
| **4**<br>**C0_AHBS_A** | W1C | CPU0 AHBS Master ECC Address Error Flag |
| **3**<br>**C0_AHBP_M** | W1C | CPU0 AHBP Master ECC Multi Bit Data Error Flag |
| **1**<br>**C0_AHBP_D** | W1C | CPU0 AHBP Master ECC Single Bit Data Error Flag |
| **0**<br>**C0_AHBP_A** | W1C | CPU0 AHBP Master ECC Address Error Flag |

### 25.6.28 SCM_MATRIX_STATUS5 (Matrix Status Register 5)

The register is POR Only.

0b - Error not occurred

1b - Error exists; write one to clear

Offset : C4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DMA0_CFG_M | DMA0_LOCKSTEP | DMA0_CFG_S | DMA0_CFG_A | 0 | | | EDC_HI | LINE_SBC | LINE_MULTI | ENC | EDC |
| W | | | | | w1c | w1c | w1c | w1c | | | | w1c | w1c | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | P2_D_M | P1_D_M | P0_D_M | 0 | P2_C_M | P1_C_M | P0_C_M | 0 | P2_D_S | P1_D_S | P0_D_S | 0 | P2_C_S | P1_C_S | P0_C_S |
| W | | w1c | w1c | w1c | | w1c | w1c | w1c | | w1c | w1c | w1c | | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_MATRIX_STATUS5 field descriptions**

| Field | Type | Description |
|---|---|---|
| **27 DMA0_CFG_M** | W1C | DMA0_CFG ECC Multi Bit Data Error Flag |
| **26 DMA0_LOCKSTEP** | W1C | DMA0 Lockstep Error Flag |
| **25 DMA0_CFG_S** | W1C | DMA0_CFG ECC Single Bit Data Error Flag |
| **24 DMA0_CFG_A** | W1C | DMA0_CFG ECC Address Error Flag |
| **20 EDC_HI** | W1C | Flash Control OCM EDC FCCU Alarm High Error Flag |
| **19 LINE_SBC** | W1C | FCCU Alarm Linebuffer Self ECC Single Bit Error Flag |
| **18 LINE_MULTI** | W1C | FCCU Alarm Linebuffer Self ECC Multi Bit Error Flag |
| **17 ENC** | W1C | FLASH Control OCM ENC FCCU Alarm Error Flag |
| **16 EDC** | W1C | Flash Control OCM EDC FCCU Alarm Error Flag |
| **14 P2_D_M** | W1C | Flash Control Port 2 Data Code Multi Error Flag |

| | | |
|---|---|---|
| **13**<br>**P1_D_M** | W1C | Flash Control Port 1 Data Code Multi Error Flag |
| **12**<br>**P0_D_M** | W1C | Flash Control Port 0 Data Code Multi Error Flag |
| **10**<br>**P2_C_M** | W1C | Flash Control Port 2 ECC Code Multi Error Flag |
| **9**<br>**P1_C_M** | W1C | Flash Control Port 1 ECC Code Multi Error Flag |
| **8**<br>**P0_C_M** | W1C | Flash Control Port 0 ECC Code Multi Error Flag |
| **6**<br>**P2_D_S** | W1C | Flash Control Port 2 Data Code Single Error Flag |
| **5**<br>**P1_D_S** | W1C | Flash Control Port 1 Data Code Single Error Flag |
| **4**<br>**P0_D_S** | W1C | Flash Control Port 0 Data Code Single Error Flag |
| **2**<br>**P2_C_S** | W1C | Flash Control Port 2 ECC Code Single Error Flag |
| **1**<br>**P1_C_S** | W1C | Flash Control Port 1 ECC Code Single Error Flag |
| **0**<br>**P0_C_S** | W1C | Flash Control Port 0 ECC Code Single Error Flag |

### 25.6.29 SCM_MATRIX_ID_STATUS0 (Matrix Master ID Status Register)

The register is POR Only. Write 0xf to clear.

Offset : C8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | SRAM1_ID | | | | SRAM0_ID | | | |
| W | | | | | | | | | w1c | | | | w1c | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAM0_S5_ID | | | 0 | | | | | MAM0_S4_ID | | | | MAM0_S3_ID | | | |
| W | w1c | | | | | | | | w1c | | | | w1c | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_MATRIX_ID_STATUS0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-20**<br>**SRAM1_ID** | W1C | SARM1 ECC Error with Which Master Access<br>0000b - CPU0<br>0010b - Reserved<br>0100b - Reserved<br>0110b - Reserved<br>1000b - HSM<br>1010b - DMA0<br>1100b - Reserved |
| **19-16**<br>**SRAM0_ID** | W1C | SARM0 ECC Error with Which Master Access<br>0000b - CPU0<br>0010b - Reserved<br>0100b - Reserved<br>0110b - Reserved<br>1000b - HSM<br>1010b - DMA0<br>1100b - Reserved |
| **15-12**<br>**MAM0_S5_ID** | W1C | MAM0_S5 Slave ECC Error with Which Master Access<br>0000b - CPU0<br>0010b - Reserved<br>0100b - Reserved<br>0110b - Reserved<br>1000b - HSM<br>1010b - DMA0<br>1100b - Reserved |
| **7-4**<br>**MAM0_S4_ID** | W1C | MAM0_S4 Slave ECC Error with Which Master Access<br>0000b - CPU0<br>0010b - Reserved<br>0100b - Reserved<br>0110b - Reserved<br>1000b - HSM<br>1010b - DMA0<br>1100b - Reserved |

| | | |
|---|---|---|
| **3-0**<br>**MAM0_S3_ID** | W1C | MAM0_S3 Slave ECC Error with Which Master Access<br>0000b - CPU0<br>0010b - Reserved<br>0100b - Reserved<br>0110b - Reserved<br>1000b - HSM<br>1010b - DMA0<br>1100b - Reserved |

### 25.6.30 SCM_MATRIX_STATUS6 (Matrix Status Register 6)

The register is POR Only.

0b - Error not occurred

1b - Error exists; write one to clear

Offset : CCh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | STALL_ERR | MAM0_S8_DS | MAM0_S5_DS | MAM0_S5_S2F | 0 | | SRAM1_DEC | SRAM0_DEC | AFCB1_MON | AFCB0_MON | STCU_ST | DCM_SCAN | 0 | NON_USER | SCM_CRC | SCG_CRC |
| W | w1c | w1c | w1c | w1c | | | w1c | w1c | w1c | w1c | w1c | w1c | | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | SRAM1_CTRL_MON | SRAM0_CTRL_MON | 0 | | | | SRAM1_M | SRAM1_EDC | SRAM1_A | SRAM1_S | SAM0_M | SRAM0_EDC | SRAM0_A | SRAM0_S |
| W | | | w1c | w1c | | | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_MATRIX_STATUS6 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**STALL_ERR** | W1C | MAM Stall Error Flag |
| **30**<br>**MAM0_S8_DS** | W1C | MAM0_S8 Downsize Monitor Error Flag |

| | | |
|---|---|---|
| **29**<br>**MAM0_S5_DS** | W1C | MAM0_S5 Downsize Monitor Error Flag |
| **28**<br>**MAM0_S5_S2F** | W1C | MAM0_S5 Slow to Fast Gasket Monitor Error Flag |
| **25**<br>**SRAM1_DEC** | W1C | SARM1 Decode Monitor Error Flag |
| **24**<br>**SRAM0_DEC** | W1C | SARM0 Decode Monitor Error Flag |
| **23**<br>**AFCB1_MON** | W1C | AFCB0 Monitor Error Flag |
| **22**<br>**AFCB0_MON** | W1C | AFCB1 Monitor Error Flag |
| **21**<br>**STCU_ST** | W1C | STCU Self Test Monitor Error Flag |
| **20**<br>**DCM_SCAN** | W1C | FCM Scan ECC Error Flag |
| **18**<br>**NON_USER** | W1C | STCU Nonuser Error Flag |
| **17**<br>**SCM_CRC** | W1C | SCM CRC Error Flag |
| **16**<br>**SCG_CRC** | W1C | SCG CRC Error Flag |
| **13**<br>**SRAM1_CTRL_MON** | W1C | SRAM1 Control Monitor Error Flag |
| **12**<br>**SRAM0_CTRL_MON** | W1C | SRAM0 Control Monitor Error Flag |
| **7**<br>**SRAM1_M** | W1C | SRAM1 Multi Bit Data Error Flag |
| **6**<br>**SRAM1_EDC** | W1C | SRAM1 EDC Error Flag |
| **5**<br>**SRAM1_A** | W1C | SRAM1 Address Error Flag |
| **4**<br>**SRAM1_S** | W1C | SRAM1 Single Bit Data Error Flag |
| **3**<br>**SAM0_M** | W1C | SRAM0 Multi Bit Data Error Flag |

| | | |
|---|---|---|
| **2**<br>**SRAM0_EDC** | W1C | SRAM0 EDC Error Flag |
| **1**<br>**SRAM0_A** | W1C | SRAM0 Address Error Flag |
| **0**<br>**SRAM0_S** | W1C | SRAM0 Single Bit Data Error Flag |

### 25.6.31 SCM_MATRIX_STATUS7 (Matrix Status Register 7)

The register is POR Only.

0b - Error not occurred

1b - Error exists; write one to clear

Offset : D0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | HSM_IRAM_M | HSM_IRAM_S | HSM_DRAM_M | HSM_DRAM_S | HSM_WDOG | 0 | MAM0_ERR | 0 | FOSC_ERR | PLL1_ERR | PLL0_ERR | |
| W | | | | | w1c | w1c | w1c | w1c | w1c | | w1c | | w1c | w1c | w1c | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_MATRIX_STATUS7 field descriptions**

| Field | Type | Description |
|---|---|---|
| **11**<br>**HSM_IRAM_M** | W1C | HSM IRAM Multi Bit Data Error Flag |
| **10**<br>**HSM_IRAM_S** | W1C | HSM IRAM Single Bit Data Error Flag |
| **9**<br>**HSM_DRAM_M** | W1C | HSM DRAM Multi Bit Data Error Flag |
| **8**<br>**HSM_DRAM_S** | W1C | HSM DRAM Single Bit Data Error Flag |

| | | |
|---|---|---|
| **7**<br>**HSM_WDOG** | W1C | HSM WDOG Request |
| **4**<br>**MAM0_ERR** | W1C | MAM0 Timeout Error Flag |
| **2**<br>**FOSC_ERR** | W1C | FOSC Loss of Clock Error Flag |
| **1**<br>**PLL1_ERR** | W1C | PLL1 Loss of Clock Error Flag |
| **0**<br>**PLL0_ERR** | W1C | PLL0 Loss of Clock Error Flag |

### 25.6.32 SCM_TPU_GTBCM (TPU Global Time Base Control Mask Register)

Offset : F0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TPU_TRGSEL | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | TPU_GTBCM | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_TPU_GTBCM field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**TPU_TRGSEL** | RW | TPU Channel [31:24] Source Select<br>0b - TPU INT CH_X Select TPU_CH_X PAD<br>1b - TPUINT_CH_X select TRGSEL5 0utput |
| **3-0**<br>**TPU_GTBCM** | RW | TPU GTBC Mask is corresponding to channel trigger of TSTMP1<br>0b - TPU GTBC not be affected by channel trigger of TSTMP1<br>1b - TPU GTBC is written to 1 when corresponding channel trigger of STMP1 is active |

### 25.6.33 SCM_FTU_GTBCM (FTU Global Time Base Control Mask Register)

Offset : F4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FTU7_GTBCM | | | | FTU6_GTBCM | | | | FTU5_GTBCM | | | | FTU4_GTBCM | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FTU3_GTBCM | | | | FTU2_GTBCM | | | | FTU1_GTBCM | | | | FTU0_GTBCM | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_FTU_GTBCM field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-28**<br>**FTU7_GTBCM** | RW | FTU7 GTBC Mask is corresponding to channel trigger of TSTMP1<br>0b - FTU7 GTBC not be affected by channel trigger of TSTMP1<br>1b - FTU7 GTBC is written to 1 when corresponding channel trigger of STMP1 is active |
| **27-24**<br>**FTU6_GTBCM** | RW | FTU6 GTBC Mask is corresponding to channel trigger of TSTMP1<br>0b - FTU6 GTBC not be affected by channel trigger of TSTMP1<br>1b - FTU6 GTBC is written to 1 when corresponding channel trigger of STMP1 is active |
| **23-20**<br>**FTU5_GTBCM** | RW | FTU5 GTBC Mask is corresponding to channel trigger of TSTMP1<br>0b - FTU5 GTBC not be affected by channel trigger of TSTMP1<br>1b - FTU5 GTBC is written to 1 when corresponding channel trigger of STMP1 is active |
| **19-16**<br>**FTU4_GTBCM** | RW | FTU4 GTBC Mask is corresponding to channel trigger of TSTMP1<br>0b - FTU4 GTBC not be affected by channel trigger of TSTMP1<br>1b - FTU4 GTBC is written to 1 when corresponding channel trigger of STMP1 is active |
| **15-12**<br>**FTU3_GTBCM** | RW | FTU3 GTBC Mask is corresponding to channel trigger of TSTMP1<br>0b - FTU3 GTBC not be affected by channel trigger of TSTMP1<br>1b - FTU3 GTBC is written to 1 when corresponding channel trigger of STMP1 is active |
| **11-8**<br>**FTU2_GTBCM** | RW | FTU2 GTBC Mask is corresponding to channel trigger of TSTMP1<br>0b - FTU2 GTBC not be affected by channel trigger of TSTMP1<br>1b - FTU2 GTBC is written to 1 when corresponding channel trigger of STMP1 is active |

| | | |
|---|---|---|
| **7-4**<br>**FTU1_GTBCM** | RW | FTU1 GTBC Mask is corresponding to channel trigger of TSTMP1<br>0b - FTU1 GTBC not be affected by channel trigger of TSTMP1<br>1b - FTU1 GTBC is written to 1 when corresponding channel trigger of STMP1 is active |
| **3-0**<br>**FTU0_GTBCM** | RW | FTU0 GTBC Mask is corresponding to channel trigger of TSTMP1<br>0b - FTU0 GTBC not be affected by channel trigger of TSTMP1<br>1b - FTU0 GTBC is written to 1 when corresponding channel trigger of STMP1 is active |

### 25.6.34 SCM_SYSAP_MDO (SYSAP MDO Register)

Offset : 100h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SYSAP_MDO | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SYSAP_MDO | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_SYSAP_MDO field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**SYSAP_MDO** | RW | Mailbox Data OUT, output from device to debug |

### 25.6.35 SCM_SYSAP_MDI (SYSAP MDI Register)

Offset : 104h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SYSAP_MDI | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SYSAP_MDI | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_SYSAP_MDI field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 SYSAP_MDI** | RO | Mailbox Data IN, input from debug to device |

### 25.6.36 SCM_SYSAP_CTRL (SYSAP Control Register)

Offset : 108h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | STATUS2_REG_SEL | | | NEW_MDI_AVAILABLE | 0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | CPU0_HOLD_IN_WAIT | CPU0_HOLD_IN_RESET | SYSTEM_RESET_REQ | CPU_DEBUG_RESTART | 0 | CPU_DEBUG_REQ | | | | | | | | DEBUG_DIS | FLASH_MASS_ERASE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_SYSAP_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **20-18**<br>**STATUS2_REG_SEL** | RO | Status2 Register Selection<br>Can be written even system secure.<br>000b - Select UID0<br>001b - Select UID1<br>010b - Select UID2<br>011b - Select UID3<br>100b - NVR version<br>101-111b - Reserved |
| **17**<br>**NEW_MDI_AVAILABLE** | RO | New Mailbox Data Input Available<br>Device read MDI register will automatically clear the bit. |
| **14**<br>**CPU0_HOLD_IN_WAIT** | RO | CPU0 Hold in Wait<br>Control if hold CPU0 in wait at the end of reset sequence.<br>0b - CPU0 is not in wait at the end of reset sequence<br>1b - CPU0 is in wait at the end of reset sequence |
| **13**<br>**CPU0_HOLD_IN_RESET** | RO | CPU0 Hold in Reset<br>Control if hold CPU0 in reset at the end of reset sequence.<br>0b - CPU0 is not in reset at the end of reset sequence<br>1b - CPU0 is in reset at the end of reset sequence |
| **12**<br>**SYSTEM_RESET_REQ** | RO | System Reset Request<br>Set to force a system reset; clear the bit to release system from reset.<br>0b - System reset request not assert<br>1b - System reset reguest assert |
| **11**<br>**CPU_DEBUG_RESTART** | RO | CPU Debug Request<br>Set to force all CPU debug restarted, CPU to exit halt state.<br>0b - CPU debug not restarted<br>1b - CPU debug restarted |
| **9-2**<br>**CPU_DEBUG_REQ** | RO | CPU Debug Request<br>Set to force core to halt, if core is in stop mode.<br>This bit can be used to wakeup the core and transition to halt state.<br>02h - CPU0 debug request |
| **1**<br>**DEBUG_DIS** | RO | Debug Disable<br>Set to disable the debug; clear to allow debug operation.<br>0b - Debug not disabled<br>1b - Debug disabled |
| **0**<br>**FLASH_MASS_ERASE** | RO | Flash Mass Erase<br>Set to Trigger Mass Erase; clear by hardware after mass erase is done.<br>If mass erase is disabled, the erase request does not occur, and the bit keeps asserted until the next system reset.<br>0b - Mass erase not triggered<br>1b - Mass erase triggered |

### 25.6.37 SCM_SUBSYS_PCC (Subsystem PCC Register)

Offset : 110h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WPB_LOCK | WPB | | | 0 | | CLKEN_TPU | CLKEN_HSM_ENGINE | CLKEN_SUBSYS | 0 | | | | | | SWRST |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | WDG_RSTEN | 0 | | | STOP_REQ |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_SUBSYS_PCC field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**WPB_LOCK** | RW | Write Permission Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**WPB** | RW | Write Permission<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |
| **25**<br>**CLKEN_TPU** | RW | TPU Clock Enable<br>Note: Only available in HSM mode.<br>0b - TPU clock disable<br>1b - TPU clock enable |
| **24**<br>**CLKEN_HSM_ENGINE** | RW | HSM Clock Enable<br>Note: Only available in FlexCore mode.<br>0b - HSM clock disable<br>1b - HSM clock enable |

| | | |
|---|---|---|
| **23**<br>**CLKEN_SUBSYS** | RW | Subsystem Clock Enable<br>0b - Subsystem clock disable<br>1b - Subsystem clock enable |
| **16**<br>**SWRST** | RW | Subsystem Software Reset<br>0b - Not generate subsystem software reset<br>1b - Generate subsystem software reset |
| **4**<br>**WDG_RSTEN** | RW | Subsystem WDG Reset Enable<br>0b - Subsystem WDG reset disable<br>1b - Subsystem WDG reset enable |
| **0**<br>**STOP_REQ** | RW | Subsystem Stop Mode Request<br>0b - Not assert subsystem stop mode request<br>1b - Assert subsystem stop mode request |

### 25.6.38 SCM_SUBSYS_STATUS (Subsystem Status Register)

Offset : 114h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SUBSYS_ERR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | CHIP_IS_FT | CHIP_IS_VIRGIN | TPU_CLK_EN | SUBSYS_RST | SUBSYS_ERR_INT | WDG_RST | INIT_DONE | ISP_TOGGLE | SUBSYS_SLEEP | STOP_ACK |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_SUBSYS_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-16**<br>**SUBSYS_ERR** | RO | Subsystem Error Report |
| **9**<br>**CHIP_IS_FT** | RO | SEC DC Lifecycle FT<br>0b - Chip is not in FT status<br>1b - Chip is in FT status |

| | | |
|---|---|---|
| **8**<br>**CHIP_IS_VIRGIN** | RO | SEC DC Lifecycle Virgin<br>0b - Chip is not in Virgin status<br>1b - Chip is in Virgin status |
| **7**<br>**TPU_CLK_EN** | RO | TPU Clock Enable<br>0b - TPU clock disable<br>1b - TPU clock enable |
| **6**<br>**SUBSYS_RST** | RO | Subsystem Reset State<br>0b - Not subsystem reset state<br>1b - Subsystem reset state |
| **5**<br>**SUBSYS_ERR_INT** | RO | Subsystem Error Interrupt<br>0b - Not subystem error interrupt<br>1b - Subsystem error interrupt |
| **4**<br>**WDG_RST** | RO | Subsystem WDOG Generated Reset Flag<br>0b - Not subsystem WDOG generated reset<br>1b - Subsystem WDOG generated reset |
| **3**<br>**INIT_DONE** | RO | Subsystem Initial Done Flag<br>0b - Subsystem Initial not done<br>1b - Subsystem Initial done |
| **2**<br>**ISP_TOGGLE** | RO | ISP Pin Toggle Flag<br>0b - ISP Pin not toggled<br>1b - ISP Pin toggled |
| **1**<br>**SUBSYS_SLEEP** | RO | Subsystem Sleep State<br>0b - Not subsystem sleep state<br>1b - Subsystem sleep state |
| **0**<br>**STOP_ACK** | RO | Subsystem Stop Mode Ack<br>0b - Stop acknowledge is not asserted<br>1b - Stop acknowledge is asserted (the module is in Stop mode) |

### 25.6.39 SCM_MDO_FLAG (Mailbox Data Output Flag Register)

Offset : 118h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | MDO_ |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | FLAG |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_MDO_FLAG field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**MDO_FLAG** | RW | Mailbox data output from device to debug flag. |

### 25.6.40 SCM_MASTER_HALT_REQ (Master Halt Request Register)

Offset : 120h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WPB_ | WPB | | | 0 | | | | | | | | | | | DMA0 |
| W | LOCK | | | | | | | | | | | | | | | _REQ |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | HSM_ |
| W | | | | | | | | | | | | | | | | REQ |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_MASTER_HALT_REQ field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**WPB_LOCK** | RW | Write Permission Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**WPB** | RW | Write Permission<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |
| **16**<br>**DMA0_REQ** | RW | DMA0 Halt Request<br>0b - DMA0 halt request not assert<br>1b - DMA0 halt request assert |

| | | |
|---|---|---|
| **0**<br>**HSM_REQ** | RW | HSM Halt Request<br>0b - HSM halt request not assert<br>1b - HSM halt request assert |

### 25.6.41 SCM_MASTER_HALT_ACK (Master Halt ACK Register)

Offset : 124h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | DMA0_ACK | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | HSM_ACK | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_MASTER_HALT_ACK field descriptions**

| Field | Type | Description |
|---|---|---|
| **18-16**<br>**DMA0_ACK** | RO | DMA0 Halted State<br>0b - Not DMA0 Halted State<br>1b - DMA0 Halted State |
| **2-0**<br>**HSM_ACK** | RO | HSM Halted State<br>0b - Not HSM Halted State<br>1b - HSM Halted State |

### 25.6.42 SCM_INT_ROUTER_NMI (NMI Interrupt Router Register)

Offset : 1FCh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | C0_EN | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | LOCK |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_INT_ROUTER_NMI field descriptions**

| Field | Type | Description |
|---|---|---|
| **24**<br>**C0_EN** | RW | CPU0 NMI Interrupt Router Enable<br>0b - CPU0 NMI interrupt router disable<br>1b - CPU0 NMI interrupt router enable |
| **0**<br>**LOCK** | RW | 0b - Unlock Cx_EN; can be written<br>1b - Lock Cx_EN; can not be written |

### 25.6.43 SCM_CRCCSR (CRC Control Status Register)

Offset : 800h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | DONE | ERR | BUY | EOEN | CHKEN | TRGEN | GEN |
| W | | | | | | | | | | w1c | w1c | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_CRCCSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **6**<br>**DONE** | W1C | CRC Done Flag<br>0b - CRC Done not occurred<br>1b - CRC Done occurred |
| **5**<br>**ERR** | W1C | CRC Error Flag<br>0b - CRC Error not occurred<br>1b - CRC Error occurred |
| **4**<br>**BUY** | RO | CRC Busy Flag<br>0b - CRC is not busy<br>1b - CRC is busy |

| | | |
|---|---|---|
| **3**<br>**EOEN** | RW | CRC Error Out Enable<br>0b - CRC Error Out disable<br>1b - CRC Error Out enable |
| **2**<br>**CHKEN** | RW | CRC Check Enable<br>0b - CRC check disable<br>1b - CRC check enable |
| **1**<br>**TRGEN** | RW | CRC Trigger Enable<br>0b - CRC trigger disable<br>1b - CRC trigger enable |
| **0**<br>**GEN** | RW | CRC Software Generation Enable<br>0b - CRC Software generation disable<br>1b - CRC Software generation enable |

### 25.6.44 SCM_CRCRES (CRC Result Register)

Offset : 804h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RESULT | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RESULT | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCM_CRCRES field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**RESULT** | RW | CRC Result Value<br>This field stores the CRC result generated.<br>When CRCCSR[CHKEN] is set, reading this register gets the CRC result. It's a read-only register.<br>When CRCCSR[CHKEN] is cleared, writing this register can configure the golden CRC value. Reading this register can get the golden CRC value. |

# Chapter 26 Matrix Access Monitor (MAM)

## 26.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 26.2 Chip-specific Information

**Table 58. MAM information in SOC**

| Name | MASTER_NUM | SLAVE_NUM | Block Number |
|---|---|---|---|
| MAM0 | 4 | 9 | 371 |

**Table 59. MAM0 master information**

| Name | Index |
|---|---|
| CPU0_AXIM | 0 |
| CPU0_AHBP | 1 |
| DMA0 | 2 |
| HSM | 3 |

**Table 60. MAM0 slave information**

| MAM Index | Name | Start Address | End Address | Block | Block Size | Block Index | Total |
|---|---|---|---|---|---|---|---|
| MAM0(S0) | ROM | 0480_0000 | 0481_FFFF | 1 | 128 KB | 0 | 128 KB |
| MAM0(S1) | FLASH0(PFLASH) | 0100_0000 | 011F_FFFF | 16 | 128 KB | 1-16 | 2048 KB |
| | NVR0/1/2 | 0440_0000 | 0441_1FFF | 1 | 128 KB | 17 | 40 KB |
| | NVR_CFG | 0442_0000 | 0443_1FFF | 1 | 128 KB | 18 | 2 KB |
| MAM0(S2) | FLASH1(DFLASH) | 0400_0000 | 0401_FFFF | 16 | 8 KB | 19-34 | 128 KB |
| MAM0(S3) | SRAM0 | 2100_0000 | 2100_FFFF | 4 | 16 KB | 35-38 | 64 KB |
| MAM0(S4) | SRAM1 | 2101_0000 | 2101_7FFF | 2 | 16 KB | 39-40 | 32 KB |
| MAM0(S5) | CPU0_ITCM | 0000_0000 | 0000_7FFF | 2 | 16 KB | 41-42 | 32 KB |
| | CPU0_DTCM | 2000_0000 | 2001_FFFF | 8 | 16 KB | 43-50 | 128 KB |
| MAM0(S6) | AFCB0 | 4000_0000 | 4009_FFFF | 160 | 4 KB | 51-210 | 640 KB |
| MAM0(S7) | AFCB1 | 4040_0000 | 4049_FFFF | 160 | 4 KB | 211-370 | 640 KB |

**Table 61. MAM0 ACLRn registers information**

| Register Name | Mask | Contents |
|---|---|---|
| ACLR0 | 0xFFFFFFFF | {Block31, Block30, ···., Block1, Block0} |
| ACLR1 | 0x3807FFFF | {Block63, Block62, ···., Block33, Block32} |
| ACLR2 | 0xFEFE78C4 | Refer to the AFCB0/1 slot assignment table in order. |
| ACLR3 | 0x7FDCCDFF | Refer to the AFCB0/1 slot assignment table in order. |
| ACLR4 | 0x7AE78880 | Refer to the AFCB0/1 slot assignment table in order. |
| ACLR5 | 0x07FC1C28 | Refer to the AFCB0/1 slot assignment table in order. |
| ACLR6 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. |
| ACLR7 | 0x00000001 | Refer to the AFCB0/1 slot assignment table in order. |
| ACLR8 | 0x031CC040 | Refer to the AFCB0/1 slot assignment table in order. |
| ACLR9 | 0x7C3C0000 | Refer to the AFCB0/1 slot assignment table in order. |
| ACLR10 | 0x07F80080 | Refer to the AFCB0/1 slot assignment table in order. |

Flagchip

| | | |
|---|---|---|
| ACLR11 | 0x00008000 | Refer to the AFCB0/1 slot assignment table in order. |

**Table 62. MAM0 ACRn registers information**

| Register Name | Mask | Contents | Master |
|---|---|---|---|
| ACR0 | 0xFFFFFFFF | {Block7, Block6, ⋯., Block1, Block0} | CPU0_AXIM |
| ACR1 | 0xFFFFFFFF | {Block15, Block14, ⋯., Block9, Block8} | |
| ACR2-5 | 0xFFFFFFFF | Arranged in order from Block16 to Block47 | |
| ACR6 | 0x00000FFF | Arranged in order from Block48 to Block55 | |
| ACR7-46 | 0x00000000 | Reserved | |
| ACR47-53 | 0x00000000 | Reserved | CPU0_AHBP |
| ACR54 | 0x00FFF000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR55 | 0xFF000F00 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR56 | 0x0FFFF000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR57 | 0xFFFFFFF0 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR58 | 0xFFFFFFF0 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR59 | 0xFFFFFFFF | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR60 | 0xFF00FF0F | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR61 | 0xFF0FFF00 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR62 | 0x0FFFFFFF | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR63 | 0xF0000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR64 | 0xF000F000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR65 | 0xFFF00FFF | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR66 | 0x0FFFF0F0 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR67 | 0x00F0F000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR68 | 0x000FFF00 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR69 | 0xFFFFFF00 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR70 | 0x00000FFF | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR71 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR72 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR73 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR74 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR75 | 0x0000000F | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR76 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR77 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR78 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR79 | 0x0F000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR80 | 0xFF000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR81 | 0x000FFF00 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR82 | 0x000000FF | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR83 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR84 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |

Flagchip

| Register Name | Mask | Contents | Master |
|---|---|---|---|
| ACR85 | 0x00FFFF00 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR86 | 0x0FFFFF00 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR87 | 0xF0000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR88 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR89 | 0xFFFFF000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR90 | 0x00000FFF | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR91 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | CPU0_AHBP |
| ACR92 | 0xF0000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR93 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR94 | 0xFFFFFFFF | {Block7, Block6, ···., Block1, Block0} | |
| ACR95 | 0xFFFFFFFF | {Block15, Block14, ···., Block9, Block8} | |
| ACR96-99 | 0xFFFFFFFF | Arranged in order from Block16 to Block47 | |
| ACR100 | 0x00000FFF | Arranged in order from Block48 to Block55 | |
| ACR101 | 0x00FFF000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR102 | 0xFF000F00 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR103 | 0x0FFFF000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR104 | 0xFFFFFFF0 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR105 | 0xFFFFFFF0 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR106 | 0xFFFFFFFF | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR107 | 0xFF00FF0F | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR108 | 0xFF0FFF00 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR109 | 0x0FFFFFFF | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR110 | 0xF0000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR111 | 0xF000F000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR112 | 0xFFF00FFF | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR113 | 0x0FFFF0F0 | Refer to the AFCB0/1 slot assignment table in order. | DMA0 |
| ACR114 | 0x00F0F000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR115 | 0x000FFF00 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR116 | 0xFFFFFF00 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR117 | 0x00000FFF | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR118 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR119 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR120 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR121 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR122 | 0x0000000F | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR123 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR124 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR125 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR126 | 0x0F000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR127 | 0xFF000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR128 | 0x000FFF00 | Refer to the AFCB0/1 slot assignment table in order. | |

Flagchip

| Register Name | Mask | Contents | Master |
|---|---|---|---|
| ACR129 | 0x000000FF | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR130 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR131 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR132 | 0x00FFFF00 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR133 | 0x0FFFFF00 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR134 | 0xF0000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR135 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | DMA0 |
| ACR136 | 0xFFFFF000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR137 | 0x00000FFF | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR138 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR139 | 0xF0000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR140 | 0x00000000 | Refer to the AFCB0/1 slot assignment table in order. | |
| ACR141-187 | | Same as DMA; 8 blocks for each register | HSM |

## 26.3 Features

The key features of the Matrix Access Monitor（MAM) include:

- Able to assign the following four groups of chip resources:
  - Core masters
  - Non-core bus masters
  - Slave memory
  - Slave peripherals
- Each slave is partitioned according to its address range and size
- Each Master can be configured to have different access rights to each block of the Slave
- Support a locking mechanism for block access policies
- Support to break the increment burst of undefined length (INCR)
- Each Master supports watchdog count to prevent the program from being stuck
- Configurable watchdog count overflow response (interrupt or hard fault)
- Configurable watchdog counting clock

## 26.4 Overview

The MAM module provides an integrated, scalable architecture framework for access control, memory protection, and peripheral isolation. It allows software to configure Master (including core and non-core) to access the access rights of each block of Slave (memories or peripherals), namely read, write, execute, and supervisor access. At the same time, it supports locking the access policy of each block to prevent malicious programs from tampering. In addition, each Master on the MAM has an independent watchdog, which prevents the slave from continuing to occupy the bus if it does not respond for a long time.

## 26.5 Functional Description

This section describes the functional description of the MAM.

### 26.5.1 Access Control

There may be multiple slaves (Memory or peripherals) on the MAM, and the address space of each slave may be very large. Therefore, the MAM supports dividing each Slave into blocks according to a fixed size. The specific block information is shown in the above three tables. Therefore, the total number of blocks on each MAM is fixed, and we need to perform permission control on these blocks. Since there are multiple Masters on the MAM, each Master can configure the permissions of each block when the Master accesses it by itself.

For example, in FC7240, there are 4 Masters and 9 Slaves on MAM0. After 9 slaves are processed in blocks, there are a total of 371 blocks. Since each Master can configure the permissions of all blocks, an access policy must be configured for a block to be accessed. An ACR register can only be configured with 8 blocks, so each Master has 47 ACR registers to configure 371 blocks, that is, CPU0 has ACR0 – ACR46, a total of 13 registers; DMA0 has ACR94 – ACR140, a total of 371, and so on.

Each ACR controller can configure 8 blocks, and the permissions of each block are controlled by 4 bits, which are read, write, execute, and supervisor. That is, a master has been configured with block0 permission of 4'b1111, then a hard fault will be generated when it has read access, when it has write access, when it fetches instructions from the block, or when it is accessed in user mode. At this time, the Master cannot have any access to block0.

### 26.5.2 Block Access Control Policy Lock

The MAM supports locking the block after configuring the block access policy. That is, at this time, the access policy of all Masters on the MAM to the block is fixed (the configuration before locking) and cannot be changed. The lock mechanism is implemented through the ACLR register. For example, in the FC7240, MAM0 has a total of 371 Blocks, that is, a total of 4 registers (ACLR0 – ACLR11).

### 26.5.3 Watchdog Timer

Each Master on the MAM has an independent watchdog counter to monitor whether the Master occupies the bus for a long time. The user can enable the watchdog counter of the corresponding Master through the WDGCR.WDOGCTRL bit, and know whether the watchdog counter-overflows through the WDGCR.WDOGSTATUS bit. The user can configure TOCR.TIMEOUTCTRL bit to specify how the MAM responds (interrupt or hard fault) after the watchdog overflow.

In addition, users can configure the watchdog's counting clock by configuring WDGDIV[WDGDIVCTRLn], which supports a maximum of 512*16 bus clocks.

## 26.6 Register Definition

MAM base address : 4040_D000h

**Register Overview - MAM**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **000h** | MAM_MAXCFG (Matrix Configuration Register) | 32 | RW | 0400_0900h | page |
| **004h** | MAM_BSTCR (Burst Control Register) | 32 | RW | 0000_0000h | page |
| **100h** | MAM_WDGCR (Watchdog Control Register) | 32 | RW | 0000_0000h | page |
| **104h** | MAM_TOCR (Timeout Control Register) | 32 | RW | 0000_0000h | page |
| **108h** | MAM_WDGDIV (Watchdog Divider Register) | 32 | RW | 0000_0000h | page |
| **700h+n*4h** | MAM_ACLRn (Access Control Lock Register) | 32 | RW | 0000_0000h | page |

| 800h+n*4h | MAM_ACRn (Access Control Register) | 32 | RW | 0000_0000h | page |
|---|---|---|---|---|---|

### 26.6.1 MAM_MAXCFG (Matrix Configuration Register)

Offset : 000h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MASTERNUM | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SLAVENUM | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MAM_MAXCFG field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**MASTERNUM** | RO | Master Number in Matrix<br>MAM Master Number is 4. |
| **15-8**<br>**SLAVENUM** | RO | Slave Number in Matrix<br>MAM Slave Number is 9. |

### 26.6.2 MAM_BSTCR (Burst Control Register)

Offset : 004h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | INCRBREAK3 | | INCRBREAK2 | | INCRBREAK1 | | INCRBREAK0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MAM_BSTCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **7-6**<br>**INCRBREAK3** | RW | Master3 Burst (INCR) Transfer Control Register Bit<br>00b - Break into INCR4 transfer<br>01b - Break into INCR8 transfer<br>10b - Break into INCR16 transfer<br>11b - Break into INCR16 transfer |
| **5-4**<br>**INCRBREAK2** | RW | Master2 Burst (INCR) Transfer Control Register Bit<br>00b - Break into INCR4 transfer<br>01b - Break into INCR8 transfer<br>10b - Break into INCR16 transfer<br>11b - Break into INCR16 transfer |
| **3-2**<br>**INCRBREAK1** | RW | Master1 Burst (INCR) Transfer Control Register Bit<br>00b - Break into INCR4 transfer<br>01b - Break into INCR8 transfer<br>10b - Break into INCR16 transfer<br>11b - Break into INCR16 transfer |
| **1-0**<br>**INCRBREAK0** | RW | Master0 Burst (INCR) Transfer Control Register Bit<br>00b - Break into INCR4 transfer<br>01b - Break into INCR8 transfer<br>10b - Break into INCR16 transfer<br>11b - Break into INCR16 transfer |

### 26.6.3 MAM_WDGCR (Watchdog Control Register)

Offset : 100h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | WDOGSTATUS3 | WDOGSTATUS2 | WDOGSTATUS1 | WDOGSTATUS0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | WDOGCTRL3 | WDOGCTRL2 | WDOGCTRL1 | WDOGCTRL0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MAM_WDGCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **19**<br>**WDOGSTATUS3** | RO | Master3 WDOG Status Bit<br>0b - WDOG counter does not overflow<br>1b - WDOG counter overflows |
| **18**<br>**WDOGSTATUS2** | RO | Master2 WDOG Status Bit<br>0b - WDOG counter does not overflow<br>1b - WDOG counter overflowing |
| **17**<br>**WDOGSTATUS1** | RO | Master1 WDOG Status Bit<br>0b - WDOG counter does not overflow<br>1b - WDOG counter overflows |
| **16**<br>**WDOGSTATUS0** | RO | Master0 WDOG Status Bit<br>0b - WDOG counter does not overflow<br>1b - WDOG counter overflows |
| **3**<br>**WDOGCTRL3** | RW | Master3 WDOG Control Bit<br>0b - WDOG disable<br>1b - WDOG enable |
| **2**<br>**WDOGCTRL2** | RW | Master2 WDOG Control Bit<br>0b - WDOG disable<br>1b - WDOG enable |
| **1**<br>**WDOGCTRL1** | RW | Master1 WDOG Control Bit<br>0b - WDOG disable<br>1b - WDOG enable |
| **0**<br>**WDOGCTRL0** | RW | Master0 WDOG Control Bit<br>0b - WDOG disable<br>1b - WDOG enable |

### 26.6.4 MAM_TOCR (Timeout Control Register)

Offset : 104h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TIMEOUTCTRL3 | | TIMEOUTCTRL2 | | TIMEOUTCTRL1 | | TIMEOUTCTRL0 | |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MAM_TOCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **7-6**<br>**TIMEOUTCTRL3** | RW | Master3 WDOG Timeout Control Bit<br>00b - After the WDOG counter overflows, it does not respond<br>01b - After the WDOG counter overflows, an interrupt is generated<br>10b - After the WDOG counter overflows, a hard fault is generated<br>11b - After the WDOG counter overflows, an interrupt and a hard fault are generated |
| **5-4**<br>**TIMEOUTCTRL2** | RW | Master2 WDOG Timeout Control Bit<br>00b - After the WDOG counter overflows, it does not respond<br>01b - After the WDOG counter overflows, an interrupt is generated<br>10b - After the WDOG counter overflows, a hard fault is generated<br>12-After the WDOG counter overflows, an interrupt and a hard fault are generated |
| **3-2**<br>**TIMEOUTCTRL1** | RW | Master1 WDOG Timeout Control Bit<br>00b - After the WDOG counter overflows, it does not respond<br>01b - After the WDOG counter overflows, an interrupt is generated<br>10b - After the WDOG counter overflows, a hard fault is generated<br>11b - After the WDOG counter overflows, an interrupt and a hard fault are generated |
| **1-0**<br>**TIMEOUTCTRL0** | RW | Master0 WDOG Timeout Control Bit<br>00b - After the WDOG counter overflows, it does not respond<br>01b - After the WDOG counter overflows, an interrupt is generated<br>10b - After the WDOG counter overflows, a hard fault is generated<br>11b - After the WDOG counter overflows, an interrupt and a hard fault are generated |

### 26.6.5 MAM_WDGDIV (Watchdog Divider Register)

Offset : 108h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WDGDIVCTRL3 | | | | WDGDIVCTRL2 | | | | WDGDIVCTRL1 | | | | WDGDIVCTRL0 | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MAM_WDGDIV field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-12**<br>**WDGDIVCTRL3** | RW | Master3 WDOG Divider Control Bit<br>0000b - WDOG no division; default is 512 bus clocks<br>0001b - WDOG division, 512*2 bus clocks<br>...<br>1111b - WDOG division, 512*16 bus clocks |
| **11-8**<br>**WDGDIVCTRL2** | RW | Master2 WDOG Divider Control Bit<br>0000b - WDOG no division; default is 512 bus clocks<br>0001b - WDOG division, 512*2 bus clocks<br>...<br>1111b - WDOG division, 512*16 bus clocks |
| **7-4**<br>**WDGDIVCTRL1** | RW | Master1 WDOG Divider Control Bit<br>0000b - WDOG no division; default is 512 bus clocks<br>0001b - WDOG division, 512*2 bus clocks<br>...<br>1111b - WDOG division, 512*16 bus clocks |
| **3-0**<br>**WDGDIVCTRL0** | RW | Master0 WDOG Divider Control Bit<br>0000b - WDOG no division; default is 512 bus clocks<br>0001b - WDOG division, 512*2 bus clocks<br>...<br>1111b - WDOG division, 512*16 bus clocks |

### 26.6.6 MAM_ACLRn (Access Control Lock Register)

Offset : 700h+N*4h (N : 0 ~ 11)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LOCK BLOC K31 | LOCK BLOC K30 | LOCK BLOC K29 | LOCK BLOC K28 | LOCK BLOC K27 | LOCK BLOC K26 | LOCK BLOC K25 | LOCK BLOC K24 | LOCK BLOC K23 | LOCK BLOC K22 | LOCK BLOC K21 | LOCK BLOC K20 | LOCK BLOC K19 | LOCK BLOC K18 | LOCK BLOC K17 | LOCK BLOC K16 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LOCK BLOC K15 | LOCK BLOC K14 | LOCK BLOC K13 | LOCK BLOC K12 | LOCK BLOC K11 | LOCK BLOC K10 | LOCK BLOC K9 | LOCK BLOC K8 | LOCK BLOC K7 | LOCK BLOC K6 | LOCK BLOC K5 | LOCK BLOC K4 | LOCK BLOC K3 | LOCK BLOC K2 | LOCK BLOC K1 | LOCK BLOC K0 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MAM_ACLRn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**LOCKBLOCK31** | RW | Block31 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **30**<br>**LOCKBLOCK30** | RW | Block30 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **29**<br>**LOCKBLOCK29** | RW | Block29 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **28**<br>**LOCKBLOCK28** | RW | Block28 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **27**<br>**LOCKBLOCK27** | RW | Block27 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **26**<br>**LOCKBLOCK26** | RW | Block26 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **25**<br>**LOCKBLOCK25** | RW | Block25 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **24**<br>**LOCKBLOCK24** | RW | Block24 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **23**<br>**LOCKBLOCK23** | RW | Block23 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |

| | | |
|---|---|---|
| **22**<br>**LOCKBLOCK22** | RW | Block22 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **21**<br>**LOCKBLOCK21** | RW | Block21 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **20**<br>**LOCKBLOCK20** | RW | Block20 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **19**<br>**LOCKBLOCK19** | RW | Block19 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **18**<br>**LOCKBLOCK18** | RW | Block18 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **17**<br>**LOCKBLOCK17** | RW | Block17 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **16**<br>**LOCKBLOCK16** | RW | Block16 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **15**<br>**LOCKBLOCK15** | RW | Block15 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **14**<br>**LOCKBLOCK14** | RW | Block14 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **13**<br>**LOCKBLOCK13** | RW | Block13 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **12**<br>**LOCKBLOCK12** | RW | Block12 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **11**<br>**LOCKBLOCK11** | RW | Block11 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |

| | | |
|---|---|---|
| **10**<br>**LOCKBLOCK10** | RW | Block10 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **9**<br>**LOCKBLOCK9** | RW | Block9 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **8**<br>**LOCKBLOCK8** | RW | Block8 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **7**<br>**LOCKBLOCK7** | RW | Block7 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **6**<br>**LOCKBLOCK6** | RW | Block6 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **5**<br>**LOCKBLOCK5** | RW | Block5 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **4**<br>**LOCKBLOCK4** | RW | Block4 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **3**<br>**LOCKBLOCK3** | RW | Block3 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **2**<br>**LOCKBLOCK2** | RW | Block2 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **1**<br>**LOCKBLOCK1** | RW | Block1 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |
| **0**<br>**LOCKBLOCK0** | RW | Block0 Lock Control Bit<br>0b - Block is unlocked<br>1b - Lock the block |

### 26.6.7 MAM_ACRn (Access Control Register)

Offset : 800h+N*4h (N : 0 ~ 187)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ACCPOLICY7 | | | | ACCPOLICY6 | | | | ACCPOLICY5 | | | | ACCPOLICY4 | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | ACCPOLICY3 | | | | ACCPOLICY2 | | | | ACCPOLICY1 | | | | ACCPOLICY0 | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MAM_ACRn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-28**<br>**ACCPOLICY7** | RW | Block7 Access Control (Read, Write or Execute)<br>xxx1 - Forbid read access<br>xx1x - Forbid write access<br>x1xx - Forbid execute access<br>1xxx - Forbid user access |
| **27-24**<br>**ACCPOLICY6** | RW | Block6 Access Control (Read, Write or Execute)<br>xxx1 - Forbid read access<br>xx1x - Forbid write access<br>x1xx - Forbid execute access<br>1xxx - Forbid user access |
| **23-20**<br>**ACCPOLICY5** | RW | Block5 Access Control (Read, Write or Execute)<br>xxx1 - Forbid read access<br>xx1x - Forbid write access<br>x1xx - Forbid execute access<br>1xxx - Forbid user access |
| **19-16**<br>**ACCPOLICY4** | RW | Block4 Access Control (Read, Write or Execute)<br>xxx1 - Forbid read access<br>xx1x - Forbid write access<br>x1xx - Forbid execute access<br>1xxx - Forbid user access |
| **15-12**<br>**ACCPOLICY3** | RW | Block3 Access Control (Read, Write or Execute)<br>xxx1 - Forbid read access<br>xx1x - Forbid write access<br>x1xx - Forbid execute access<br>1xxx - Forbid user access |

| | | |
|---|---|---|
| **11-8**<br>**ACCPOLICY2** | RW | Block2 Access Control (Read, Write or Execute)<br>xxx1 - Forbid read access<br>xx1x - Forbid write access<br>x1xx - Forbid execute access<br>1xxx - Forbid user access |
| **7-4**<br>**ACCPOLICY1** | RW | Block1 Access Control (Read, Write or Execute)<br>xxx1 - Forbid read access<br>xx1x - Forbid write access<br>x1xx - Forbid execute access<br>1xxx - Forbid user access |
| **3-0**<br>**ACCPOLICY0** | RW | Block0 Access Control (Read, Write or Execute)<br>xxx1 - Forbid read access<br>xx1x - Forbid write access<br>x1xx - Forbid execute access<br>1xxx - Forbid user access |

Flagchip

# Chapter 27 Mailbox (MB)

## 27.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 27.2 Chip-specific Information

The table below lists different cores and their corresponding master IDs used in FC7240F2M.

| Core | Master ID (BIN) |
|---|---|
| CPU0 | 0000 |
| FlexCore | 1000 |

## 27.3 Features

The Mailbox has the below features:

- Implements the semaphores function and provides a mechanism to acquire "lock and unlock"
- Support multiple channel communication between cores
- Support sending short/long message
- Support multiple interrupts, which can be triggered using a request or done event from channels

## 27.4 Overview

The Mailbox (MB) provides the functionality for multiple cores to communicate and synchronize their activities. MB hardware is implemented using two structures. They are communication channels and interrupt channels, among them, there are 16 communication channels and 4 interrupt channels.

Below is the block diagram of MB.

**Figure 38. Mailbox block diagram**

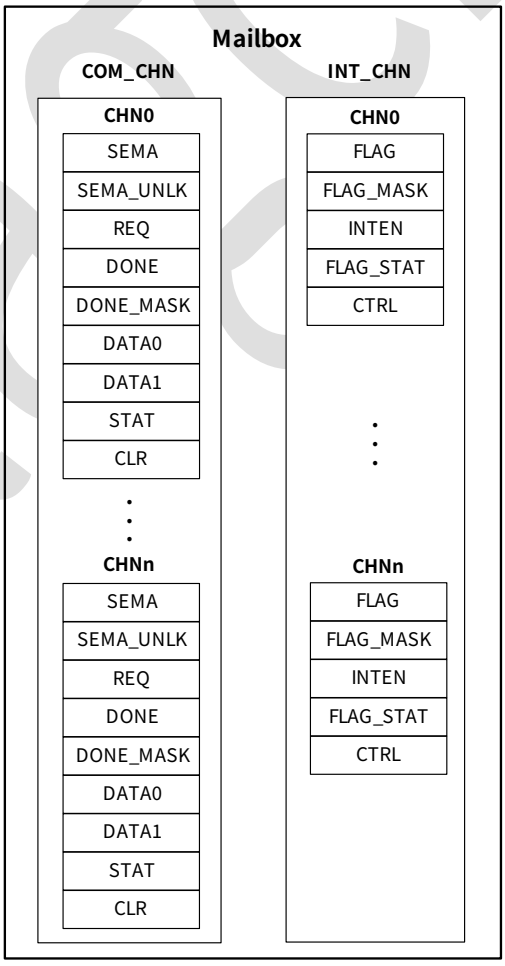
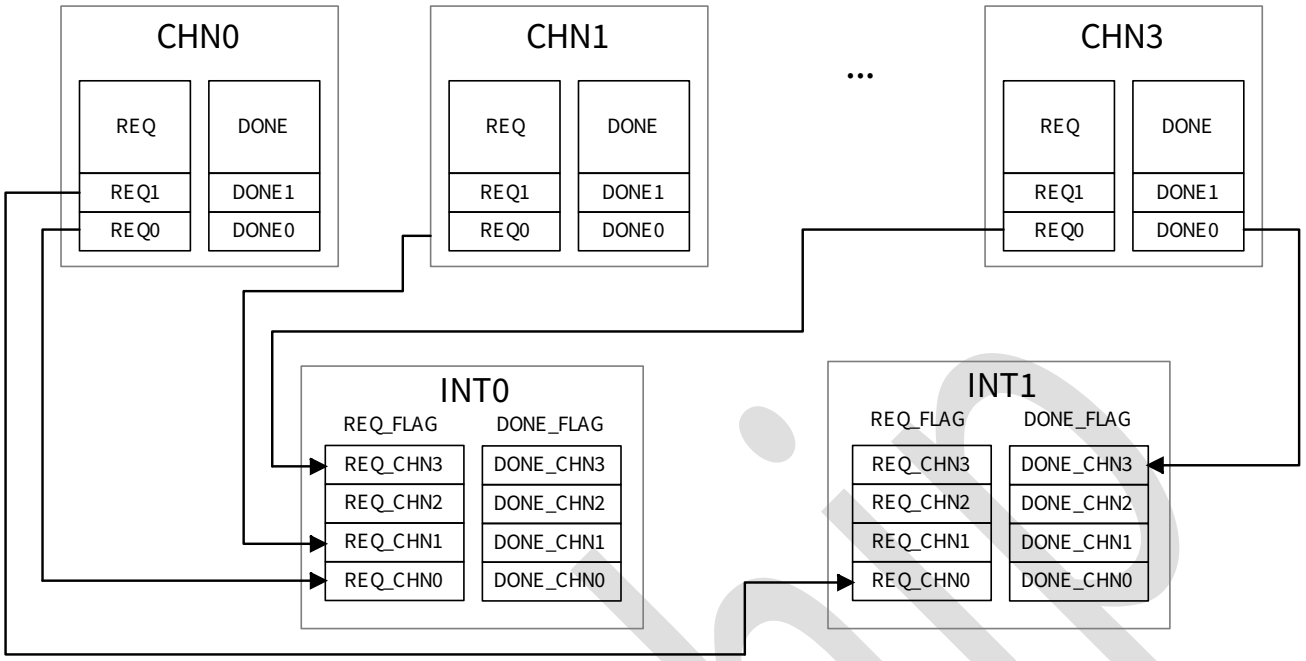


**Figure 39. Mailbox communication and interrupt channels**




## 27.5 Functional Description

### 27.5.1 Acquire Communication Channels

There are 4 communication channels in the mailbox, and the core can select any of them to obtain its use permission. The steps for obtaining channel permission are as follows:

1) Select any communication channel, assuming N, CPU0 continuously reads the SEMA register of channel N until the lock bit (bit 31) of the SEMA register is set, indicating that CPU0 obtains the channel.
2) At this time, CPU0 can communicate with other cores through the communication channel N. In addition, CPU0 or other cores can obtain the status of the channel by reading the STAT register of channel N.

### 27.5.2 Response Error

The mailbox has 4 communication channels and 2 interrupt channels. Accessing the channel register may generate a response error (hard fault) in the following cases.

1) After the communication channel N is acquired by CPU0, other cores write SEMA_UNLK, REQUEST, DATA0, DATA1, DONE_MASK these registers.
2) In the interrupt channel, access the FLG, FLG_MASK and FLG_INTEN registers while the registers are locked.

### 27.5.3 Locks for Mutual Exclusion Access

Mailbox can provide mutually exclusive access (per communication channel). The main steps are as follows:

1) CPU0 reads the SEMA register of channel N to obtain the use permission of the channel (read the SEMA register until SEMA[31]=1, indicating that the channel is obtained).
2) FlexCore reads the SEMA register of channel N to obtain the use permission of the channel (FlexCore can never obtain the channel at this time).
3) CPU0 accesses shared memory but FlexCore cannot.
4) After CPU0 completes the access, it releases Channel N by writing the DONE register 0xFC730000 of Channel N.

### 27.5.4 Short Message Passing

Mailbox can implement short message passing between cores. The main steps are as follows:

1) CPU0 ensures that REQ_MASK in FLG_MASK is turned on by reading the register of FLG_MASK of FlexCore's interrupt Channel.
2) CPU0 ensures that REQ_EN in FLG_INTEN is turned on by reading the register of FLG_INTEN of FlexCore's interrupt Channel.
3) CPU0 reads the SEMA register of channel N to obtain the use permission of the channel (read the SEMA register until SEMA[31]=1, indicating that the channel is obtained).
4) CPU0 writes the DONE_MASK register to specify which core can write to the DONE register in response to a received message.
5) CPU0 writes the data to be sent to the DATA0 and DATA1 registers.
6) CPU0 sends a request event to FlexCore's interrupt channel by writing to the REQUEST register.
7) FlexCore generates an interrupt and enters the ISR. In the ISR, read the FLG register of its own interrupt channel to determine which channel sent the req event, and clear the REQ FLAG of the FLG registers of the interrupt channel.
8) FlexCore reads DATA0 in the corresponding communication channel, and the DATA1 register receives the message.
9) FlexCore writes the DONE register in the communication channel and sends a DONE event to CPU0.
10) CPU0 generates an interrupt. In the interrupt ISR, it clears the DONE FLAG of the FLG register of the interrupt channel, and releases the communication channel.

### 27.5.5 Long Message Passing

Mailbox can implement long message passing between cores. The main steps are as follows:

1) CPU0 ensures that REQ_MASK in FLG_MASK is turned on by reading the register of FLG_MASK of FlexCore's interrupt Channel.
2) CPU0 ensures that REQ_EN in FLG_INTEN is turned on by reading the register of FLG_INTEN of FlexCore's interrupt Channel.
3) CPU0 reads the SEMA register of channel N to obtain the use permission of the channel (read the SEMA register until SEMA[31]=1, indicating that the channel is obtained).
4) CPU0 writes the DONE_MASK register to specify which core can write to the DONE register in response to a received message.
5) CPU0 writes pointer address to DATA0 and data size to DATA1.
6) CPU0 sends a REQ event to FlexCore's interrupt channel by writing to the REQUEST register.
7) FlexCore generates an interrupt and enters the ISR. In the ISR, read the FLG register of its own interrupt channel to determine which channel sent the req event, and clear the REQ FLAG of the FLG registers of the interrupt channel.
8) FlexCore reads DATA0 and DATA1 in the corresponding communication channel. And through the read information, read data from the specified memory.
9) FlexCore writes the DONE register in the communication channel and sends a DONE event to CPU0.
10) CPU0 generates an interrupt. In the interrupt ISR, it clears the DONE FLAG of the FLG register of the interrupt channel, and releases the communication channel.

Flagchip

## 27.6 Register Definition

MB base address : 4005_8000h

**Register Overview - MB**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h+n*30h** | MB_CCn_SEMA (Communication Channel Smeaphore Register) | 32 | RO | 0000_0000h | page |
| **4h+n*30h** | MB_CCn_SEMA_UNLK (Communication Channel Smeaphore Unlock Register) | 32 | RW | 0000_0000h | page |
| **8h+n*30h** | MB_CCn_REQUEST (Communication Channel Request Register) | 32 | WO | 0000_0000h | page |
| **Ch+n*30h** | MB_CCn_DONE (Communication Channel Done Register) | 32 | RW | 0000_0000h | page |
| **10h+n*30h** | MB_CCn_DONE_MASK (Communication Channel Done Mask Register) | 32 | RW | 0000_0000h | page |
| **14h+n*30h** | MB_CCn_DATA0 (Communication Channel Data Register0) | 32 | RW | 0000_0000h | page |
| **18h+n*30h** | MB_CCn_DATA1 (Communication Channel Data Register1) | 32 | RW | 0000_0000h | page |
| **1Ch+n*30h** | MB_CCn_STAT (Communication Channel Status Register) | 32 | RO | 0000_0001h | page |
| **20h+n*30h** | MB_CCn_CLR (Communication Channel Clear Register) | 32 | WO | 0000_0000h | page |
| **800h+n*20h** | MB_INTn_FLG (Interrupt Channel Flag Register) | 32 | W1C | 0000_0000h | page |
| **804h+n*20h** | MB_INTn_FLG_MASK (Interrupt Channel Flag Mask Register) | 32 | RW | 0000_0000h | page |
| **808h+n*20h** | MB_INTn_INTEN (Interrupt Channel Interrupt Enable Register) | 32 | RW | 0000_0000h | page |
| **80Ch+n*20h** | MB_INTn_FLG_STAT (Interrupt Channel Flag Status Register) | 32 | RO | 0000_0000h | page |
| **810h+n*20h** | MB_INTn_CTRL (Interrupt Channel Control Register) | 32 | RW | 0000_0007h | page |

### 27.6.1 MB_CCn_SEMA (Communication Channel Smeaphore Register)

Offset : 0h+N*30h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LOCK | 0 | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | LOCK_MASTER_ID | | | | 0 | | LOCK_MASTER_SEC | LOCK_MASTER_SUPERVISOR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MB_CCn_SEMA field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**LOCK** | RO | Channel Lock Acquisition<br>When reading 1, this bit will be cleared.<br>0b - The read did not acquire the lock<br>1b - The read acquires the lock |
| **7-4**<br>**LOCK_MASTER_ID** | RO | The master ID of the core that currently obtains the channel.<br>For specific master IDs, see Chip-specific Information. |
| **1**<br>**LOCK_MASTER_SEC** | RO | The master security information of the core that currently obtains the channel.<br>0b - Secure<br>1b - Non-secure |
| **0**<br>**LOCK_MASTER_SUPERVISOR** | RO | The master processing mode of the core that currently obtains the channel.<br>0b - Normal access<br>1b - Privileged access |

### 27.6.2 MB_CCn_SEMA_UNLK (Communication Channel Smeaphore Unlock Register)

Offset : 4h+N*30h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | AUTO | AUTO |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | _CLEAR_EN1 | _CLEAR_EN0 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MB_CCn_SEMA_UNLK field descriptions**

| Field | Type | Description |
|---|---|---|
| **1**<br>**AUTO_CLEAR_EN1** | RW | Automatically clear the channel lock enable bit (INT1).<br>0b - Disable<br>1b - Enable |
| **0**<br>**AUTO_CLEAR_EN0** | RW | Automatically clear the channel lock enable bit (INT0).<br>0b - Disable<br>1b - Enable |

### 27.6.3 MB_CCn_REQUEST (Communication Channel Request Register)

Offset : 8h+N*30h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | 0 | 0 |
| W | | | | | | | | | | | | | | | REQ1 | REQ0 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MB_CCn_REQUEST field descriptions**

| Field | Type | Description |
|---|---|---|
| **1**<br>**REQ1** | WO | Issue a Request Event to INT1.<br>0b - No effect<br>1b - Generate an event |
| **0**<br>**REQ0** | WO | Issue a Request Event to INT0.<br>0b - No effect<br>1b - Generate an event |

### 27.6.4 MB_CCn_DONE (Communication Channel Done Register)

Note: The CPU that acquires the channel N can write 0xFC730000 to this register to release the channel N.

Offset : Ch+N*30h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | DONE1 | DONE0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MB_CCn_DONE field descriptions**

| Field | Type | Description |
|---|---|---|
| **1**<br>**DONE1** | RW | Issue a Done Event to INT1<br>0b - No effect<br>1b - Generate an event |
| **0**<br>**DONE0** | RW | Issue a Done Event to INT0<br>0b - No effect<br>1b - Generate an event |

### 27.6.5 MB_CCn_DONE_MASK (Communication Channel Done Mask Register)

Offset : 10h+N*30h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | DONE_MASTER_ID | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | DONE_MASK1 | DONE_MASK0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MB_CCn_DONE_MASK field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16**<br>**DONE_MASTER_ID** | RW | Configure the master ID of the core that generates a done event.<br>For specific master IDs, see Chip-specific Information. |
| **1**<br>**DONE_MASK1** | RW | Mask INT1 Done Event<br>0b - Mask<br>1b - Unmask |
| **0**<br>**DONE_MASK0** | RW | Mask INT0 Done Event<br>0b - Mask<br>1b - Unmask |

### 27.6.6 MB_CCn_DATA0 (Communication Channel Data Register0)

Offset : 14h+N*30h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA0 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA0 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MB_CCn_DATA0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**DATA0** | RW | For sending and receiving data. |

### 27.6.7 MB_CCn_DATA1 (Communication Channel Data Register1)

Offset : 18h+N*30h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA1 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA1 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MB_CCn_DATA1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0<br>DATA1** | RW | For sending and receiving data. |

### 27.6.8 MB_CCn_STAT (Communication Channel Status Register)

Offset : 1Ch+N*30h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CURRENT_LOCK_STATUS | 0 | | | | | | | CURRENT_CPU_STATUS | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CURRENT_CPU_STATUS | | | | | | | | CURRENT_LOCK_MASTER_ID | | | | 0 | | CURRENT_LOCK_MASTER_SEC | CURRENT_LOCK_MASTER_SUPERVISOR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**MB_CCn_STAT field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**CURRENT_LOCK_STATUS** | RO | Channel Status<br>0b - The current channel is in the unlock state<br>1b - The current channel is in lock state |
| **23-8**<br>**CURRENT_CPU_STATUS** | RO | The CPU number of the currently obtained channel. |
| **7-4**<br>**CURRENT_LOCK_MASTER_ID** | RO | The master ID information of the currently obtained channel.<br>For specific master IDs, see Chip-specific Information. |
| **1**<br>**CURRENT_LOCK_MASTER_SEC** | RO | The master security information of the currently obtained channel.<br>0b - Secure<br>1b - Non-secure |
| **0**<br>**CURRENT_LOCK_MASTER_SUPERVISOR** | RO | The master processing mode of the currently obtained channel.<br>0b - Normal access<br>1b - Privileged access |

### 27.6.9 MB_CCn_CLR (Communication Channel Clear Register)

Offset : 20h+N*30h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | SOFTWARE_CLEAR_LOCK | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | SOFTWARE_CLEAR_LOCK | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MB_CCn_CLR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**SOFTWARE_CLEAR_LOCK** | WO | Software clears channel lock by writing key (0xFC350000). |

### 27.6.10 MB_INTn_FLG (Interrupt Channel Flag Register)

Offset : 800h+N*20h (N : 0 ~ 1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | DONE_FLAG | | | |
| W | | | | | | | | | | | | | w1c | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | REQ_FLAG | | | |
| W | | | | | | | | | | | | | w1c | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MB_INTn_FLG field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16**<br>**DONE_FLAG** | W1C | Done event from the communication channel.<br>Bit16 represents communication channel 0, and so on.<br>0b - No event<br>1b - Done event generation flag<br>Write 1 to clear the flag; write 0 is invalid. |
| **3-0**<br>**REQ_FLAG** | W1C | Request event from the communication channel.<br>Bit 0 represents communication channel 0, and so on.<br>0b - No event<br>1b - Request event generation<br>Write 1 to clear the flag; write 0 is invalid. |

### 27.6.11 MB_INTn_FLG_MASK (Interrupt Channel Flag Mask Register)

Offset : 804h+N*20h (N : 0 ~ 1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | DONE_FLAG_MASK | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | REQ_FLAG_MASK | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MB_INTn_FLG_MASK field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16**<br>**DONE_FLAG_MASK** | RW | Done Event Mask (disable the done event generated from the corresponding channel)<br>Bit 16 represents communication channel 0, and so on.<br>0b - Mask<br>1b - Unmask |
| **3-0**<br>**REQ_FLAG_MASK** | RW | Request Event Mask (disable the request event generated from the corresponding channel)<br>Bit 0 represents communication channel 0, and so on.<br>0b - Mask<br>1b - Unmask |

### 27.6.12 MB_INTn_INTEN (Interrupt Channel Interrupt Enable Register)

Offset : 808h+N*20h (N : 0 ~ 1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | DONE_INT_EN | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | REQ_INT_EN | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MB_INTn_INTEN field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16**<br>**DONE_INT_EN** | RW | Done Event Interrupt Enable<br>Bit 16 represents the communication channel 0, and so on.<br>0b - Disable<br>1b - Enable |
| **3-0**<br>**REQ_INT_EN** | RW | Request Event Interrupt Enable<br>Bit 0 represents the communication channel 0, and so on.<br>0b - Disable<br>1b - Enable |

### 27.6.13 MB_INTn_FLG_STAT (Interrupt Channel Flag Status Register)

Offset : 80Ch+N*20h (N : 0 ~ 1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | FLG_DONE_INT_STAT | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | FLG_REQ_INT_STAT | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MB_INTn_FLG_STAT field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16**<br>**FLG_DONE_INT_STAT** | RO | The register provides the results of MB_INTn_FLG & MB_INTn_INTEN.<br>BIT19 - BIT16 represent a done event<br>0b - Done interrupt event not occur<br>1b - Done interrupt event occurs |
| **3-0**<br>**FLG_REQ_INT_STAT** | RO | The register provides the results of MB_INTn_FLG & MB_INTn_INTEN.<br>BIT3 - BIT0 represent a request event<br>0b - Request interrupt event not occur<br>1b - Request interrupt event occurs |

### 27.6.14 MB_INTn_CTRL (Interrupt Channel Control Register)

Offset : 810h+N*20h (N : 0 ~ 1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | INTEN_LOCK | FLG_MASK_LOCK | FLG_LOCK |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

**MB_INTn_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **2**<br>**INTEN_LOCK** | RW | INTEN Lock Control Bit<br>0b - Unlock<br>1b - Lock |
| **1**<br>**FLG_MASK_LOCK** | RW | Flag Mask Lock Control Bit<br>0b - Unlock<br>1b - Lock |
| **0**<br>**FLG_LOCK** | RW | Flag Lock Control Bit<br>0b - Unlock<br>1b - Lock |

Flagchip

# Chapter 28 Flash Memory Controller (FMC)

## 28.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 28.2 Chip-specific Information

| FC7240F2M | |
|---|---|
| **PFLASH Size** | 2 MB |
| **DFLASH Size** | 128 KB |
| **OTA** | Supported |

## 28.3 Features

The following list summarizes the key features of the flash memory controller:

- Configurable read buffer and prefetch support for code bus
- ECC events report of single-bit errors and double-bit errors on every read data
- ECC (256 + 10) support for read buffer
- ECC error injection to read buffer
- ECC double-bit error suppress
- Flash array program/erase protection support
- Flash Read-While-Write (RWW) stall support
- Flash OTA function

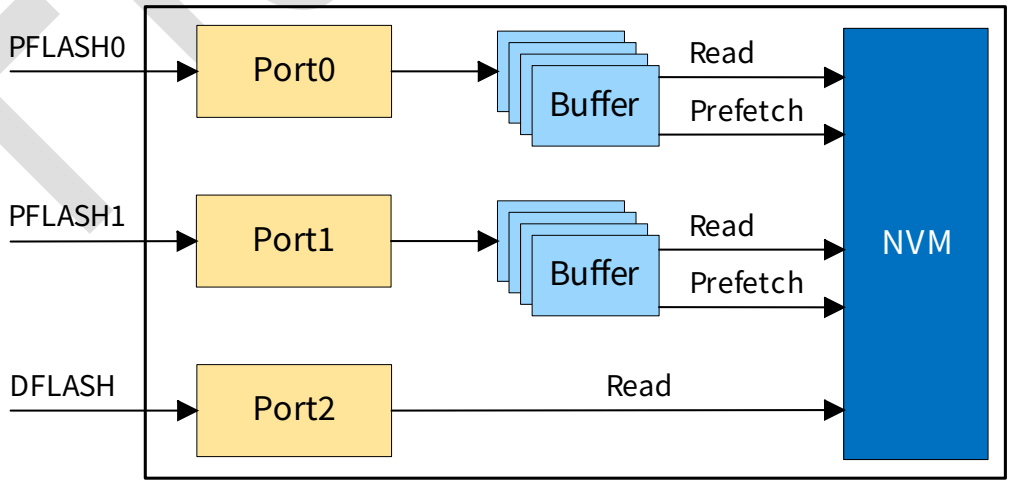
## 28.4 Overview

The Flash Memory Controller (FMC) is an intermediate module, which handles data interaction between the AHB bus and the flash memory array.

The FMC provides buffering and prefetching mechanisms to achieve data read for address hitting with zero waiting response. Read requests that miss the prefetch cache perform the actual read access to the flash array and return data to the AHB bus once completed.

The following figure shows the diagram of FMC. There is a total of three ports in the FMC. Port0 and port1 are used to access PFLASH0 and PFLASH1, while port2 is used to access DFLASH. To achieve high performance, three ports read access are independent of each other. Note that buffer and prefetch are only available for port0 and port1.

**Figure 40. FMC block diagram**




## 28.5 Functional Description

The FMC provides read and write control signals, such as a flash physical address, program/erase protection, write enable, read enable, and so on, to manage read, program and erase operation to the flash array.

### 28.5.1 Read Access

A lookup is performed first to check if the requested data is already stored in a prefetch buffer during the AHB address phase. If a buffer hit happens, target data could be returned to the AHB bus with the single-cycle response (zero wait state) from the prefetch buffer. In the event of a buffer miss, a read access to the flash array would be performed with several read wait states.

To access flash correctly, the read wait state should be configured appropriately based on operating conditions of the chip, such as CPU frequency, voltage, temperature, and so on. For the method of configuring the read wait state, refer to flash API doc.

### 28.5.2 Write Access

Program or erase sequence is initiated by first writing to FPESA_L registers.

### 28.5.3 Error Response

The FMC may generate a system bus error response in the following scenarios:

- Out-of-range program/erase address writing to FPESA_L registers.
- Improper program/erase sequence execution.
- RWW error if RWW hold is disabled.
- Double-bit ECC error detection on PFLASH and DFLASH read access if FEEC[DDBEE] = 0. For DFLASH, it is further qualified by FEEC[DES] = 0.

### 28.5.4 Read Buffers and Prefetch

For PFLASH, the flash memory controller provides buffers that are used for both prefetch and fetches. Actually, code and data fetches use the same buffers, and their distribution can be configured independently by FPAC registers.

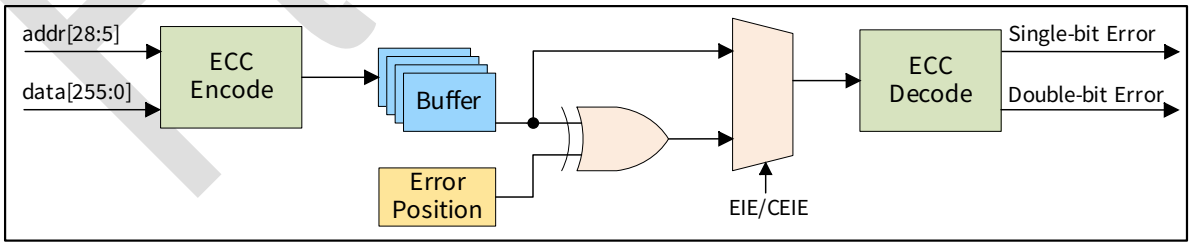
A prefetch read access may be triggered to load data into the prefetch buffer from the next double-word address upon a read transaction completion.

To enable the prefetch feature, FPAC0[xBPEN] and FPAC0[xBBEN] should both be set to 1.

### 28.5.5 Read Buffer ECC and Error Injection

ECC logic is implemented to protect data stored in the read buffer. Single-bit and double-bit faults will be reported to the Fault Collection and Safety Management Unit (FCSMU). For the detailed channel assignments, refer to Chapter FCSMU.

**Figure 41. Read buffer ECC**




ECC error injection is also supported for function safety. The user can configure FEEC[EDATAC_POS1] and FEEC[EDATAC_POS2] to set the error position.

- To inject a single-bit error, FEEC[EDATAC_POS1] and FEEC[EDATAC_POS2] should be set with the same value. Besides, only one of FEEC[EIE] and FEEC[CEIE] can be set because FEEC[EDATAC_POS1] and FEEC[EDATAC_POS2] are both valid for the ECC and Data region.

- To inject a double-bit error, FEEC[EDATAC_POS1] and FEEC[EDATAC_POS2] can be set with different values when only one of FEEC[EIE] and FEEC[CEIE] is set, or FEEC[EDATAC_POS1] and FEEC[EDATAC_POS2] with the same value when FEEC[EIE] and FEEC[CEIE] are both set.

### 28.5.6 Flash Coarse and Fine Region Protection

The FMC provides coarse and fine region protection for every flash block. If the protection function is enabled, the program or erase operation cannot be executed on the protected region, and a protection error would be indicated from flash memory.

For each flash block, fine protection covers the last 256 KB region and each fine lock bit controls one fine region whose size is 8 KB. Coarse protection is available when the block size is larger than 256 KB, which covers the other region of the block except for the last 256 KB. Each coarse lock bit takes control of the 64 KB region.

For multi-core applications, FMC also supports the set lock feature to protect the flash region for a dedicated core. For example, CPU0 can set the bit in FB_SETFLOCKn or FB_SETCLOCKn, then the master ID of CPU0 can be recorded in registers FBn_FLCKMm or FBn_CLCKMm. Once the set lock bit is asserted, only the master with the same master ID can modify the corresponding lock bit in FPELOCKn or CPELOCKn. Any modification with a different master ID to the lock bit leads to a bus fault. This means that if other masters try to write the corresponding lock bit which already recorded the CPU0 ID, a bus fault will be generated.

### 28.5.7 Flash ECC Error Report

ECC events of single-bit error and double-bit error on every read data would be reported to ERM. For the detailed channel assignment, refer to ERM doc.

### 28.5.8 Flash Read While Write Hold

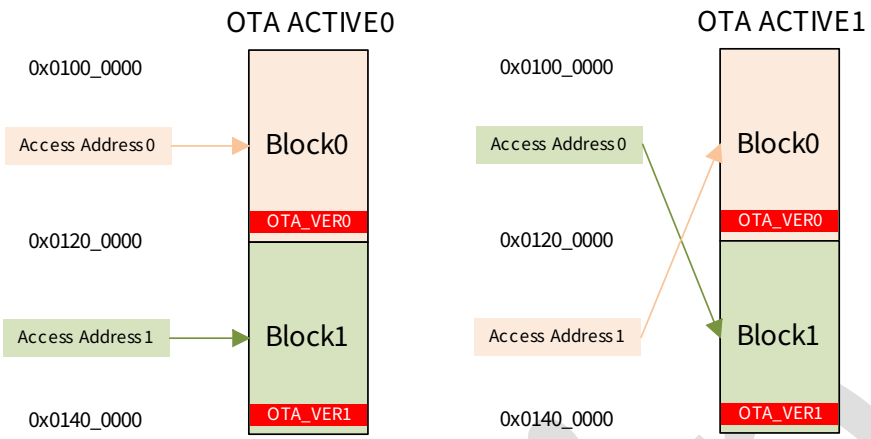
The FMC also provides RWW hold feature. In a normal case, a system bus error response would be generated when the AHB bus reads the flash block which is executing a program or erase operation. By enabling this feature, the AHB bus can keep a wait state until the write process is finished. The method to enable this feature can refer to flash API doc.

### 28.5.9 Flash OTA

OTA is supported in the FMC. If OTA is enabled, read/write access in the flash block0/1 can be swapped, which depends on OTA_CTRLx[OTA_ACTIVE]. If OTA_CTRLx[OTA_ACTIVE] is 0, the address range 0x0100_0000 - 0x0120_0000 is mapped to flash block0, while 0x0120_0000 - 0x0140_0000 is mapped to block1. If OTA_CTRLx[OTA_ACTIVE] is 1, the address range 0x0100_0000 - 0x0120_0000 is mapped to flash block1, while 0x0120_0000 - 0x0140_0000 is mapped to block0, as the below figure shows.

Flagchip

**Figure 42. OTA diagram**




The OTA_CTRL, OTA_ACT_VER, and OTA_VER_LOC registers are loaded from NVR settings after reset. Hardware will compare the OTA version value in block0 and block1 under the NVR load phase, then set the block that has a higher version value as the active block. It is recommended that the user sets bit OTA_CTRL[OTA_LOCK] to lock the OTA control registers in the initialization code.

The FC7240 supports one OTA configurations for flash block0/1.

## 28.6 Register Definition

FMC1 base address : 4001_E000h

FMC0 base address : 4001_F000h

**Register Overview - FMC**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | FMC_FAPC0 (Flash Access Port Control Register 0) | 32 | RW | 0000_0003h | page |
| **4h** | FMC_FAPC1 (Flash Access Port Control Register 1) | 32 | RW | 0000_0003h | page |
| **8h** | FMC_FAPC2 (Flash Access Port Control Register 2) | 32 | RW | 0000_0003h | page |
| **10h** | FMC_FEEC (Flash ECC Error Control Register) | 32 | RW | 0000_0000h | page |
| **18h** | FMC_FEIPC (Flash ECC Inject Position Control Register) | 32 | RW | 0000_0000h | page |
| **300h** | FMC_FPESA_L (Flash Program Erase Start Address Logical Register) | 32 | RW | 0000_0000h | page |
| **304h** | FMC_FPESA_P (Flash Program Erase Start Address Physical Register) | 32 | RO | 0000_0000h | page |
| **340h+n*4h** | FMC_FB_FPELCKn (Flash Block n Fine Program Erase Lock Register) | 32 | RW | FFFF_FFFFh | page |
| **358h** | FMC_FN_FPELCK (Flash NVR Fine Program Erase Lock Register) | 32 | RW | 0000_0001h | page |
| **35Ch+n*4h** | FMC_FB_CPELCKn (Flash Block n Coarse Program Erase Lock Register) | 32 | RW | 0FFF_FFFFh | page |
| **500h+n*4h** | FMC_OTA_CTRLn (FLASH OTA Control Register) | 32 | RW | 0000_0000h | page |
| **508h+n*4h** | FMC_OTA_VER_LOCn (FLASH OTA Version Location Register) | 32 | RO | 0000_0000h | page |

| | | | | | |
|---|---|---|---|---|---|
| **510h+n*4h** | FMC_OTA_ACT_VERn (FLASH OTA Active Version Register) | 32 | RO | 0000_0000h | page |

### 28.6.1 FMC_FAPC0 (Flash Access Port Control Register 0)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | DBPEN | CBPEN | 0 | | DBBEN | CBBEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**FMC_FAPC0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5**<br>**DBPEN** | RW | Data Bus Prefetch Enable.<br>This bit enables or disables data prefetching initiated by a read access on selected Port. Data prefetching is only valid when the buffers are enabled by setting FAPCx[DBBEN].<br>0b - Data prefetching is disabled.<br>1b - Data prefetching is enabled. |
| **4**<br>**CBPEN** | RW | Code Bus Prefetch Enable.<br>This bit enables or disables code prefetching initiated by a read access on selected Port. Code prefetching is only valid when the buffers are enabled by setting FAPCx[CBBEN].<br>0b - Code prefetching is disabled.<br>1b - Code prefetching is enabled. |
| **1**<br>**DBBEN** | RW | Data Bus Read Buffers Enable.<br>This bit enables or disables read data buffers. Buffers would also be invalidated if they are disabled.<br>0b - Read data buffers are disabled.<br>1b - Read data buffers are enabled. |

| | | |
|---|---|---|
| **0**<br>**CBBEN** | RW | Code Bus Read Buffers Enable.<br>This bit enables or disables read code buffers. Buffers would also be invalidated if they are disabled.<br>0b - Read code buffers are disabled.<br>1b - Read code buffers are enabled. |

### 28.6.2 FMC_FAPC1 (Flash Access Port Control Register 1)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | DBPEN | CBPEN | 0 | | DBBEN | CBBEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**FMC_FAPC1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5**<br>**DBPEN** | RW | Data Bus Prefetch Enable.<br>This bit enables or disables data prefetching initiated by a read access on selected Port. Data prefetching is only valid when the buffers are enabled by setting FAPCx[DBBEN].<br>0b - Data prefetching is disabled.<br>1b - Data prefetching is enabled. |
| **4**<br>**CBPEN** | RW | Code Bus Prefetch Enable.<br>This bit enables or disables code prefetching initiated by a read access on selected Port. Code prefetching is only valid when the buffers are enabled by setting FAPCx[CBBEN].<br>0b - Code prefetching is disabled.<br>1b - Code prefetching is enabled. |
| **1**<br>**DBBEN** | RW | Data Bus Read Buffers Enable.<br>This bit enables or disables read data buffers. Buffers would also be invalidated if they are disabled.<br>0b - Read data buffers are disabled.<br>1b - Read data buffers are enabled. |

| | | |
|---|---|---|
| **0**<br>**CBBEN** | RW | Code Bus Read Buffers Enable.<br>This bit enables or disables read code buffers. Buffers would also be invalidated if they are disabled.<br>0b - Read code buffers are disabled.<br>1b - Read code buffers are enabled. |

### 28.6.3 FMC_FAPC2 (Flash Access Port Control Register 2)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | DBPEN | CBPEN | 0 | | DBBEN | CBBEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**FMC_FAPC2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **5**<br>**DBPEN** | RW | Data Bus Prefetch Enable.<br>This bit enables or disables data prefetching initiated by a read access on selected Port. Data prefetching is only valid when the buffers are enabled by setting FAPCx[DBBEN].<br>0b - Data prefetching is disabled.<br>1b - Data prefetching is enabled. |
| **4**<br>**CBPEN** | RW | Code Bus Prefetch Enable.<br>This bit enables or disables code prefetching initiated by a read access on selected Port. Code prefetching is only valid when the buffers are enabled by setting FAPCx[CBBEN].<br>0b - Code prefetching is disabled.<br>1b - Code prefetching is enabled. |
| **1**<br>**DBBEN** | RW | Data Bus Read Buffers Enable.<br>This bit enables or disables read data buffers. Buffers would also be invalidated if they are disabled.<br>0b - Read data buffers are disabled.<br>1b - Read data buffers are enabled. |

| | | |
|---|---|---|
| **0**<br>**CBBEN** | RW | Code Bus Read Buffers Enable.<br>This bit enables or disables read code buffers. Buffers would also be invalidated if they are disabled.<br>0b - Read code buffers are disabled.<br>1b - Read code buffers are enabled. |

### 28.6.4 FMC_FEEC (Flash ECC Error Control Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CEIE_SET | | | | EIE_SET | | | | EDATAC_POS2 | | | | EDATAC_POS1 | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | | | | | | | | DDBEE | 0 | | | | | | DES |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_FEEC field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-28**<br>**CEIE_SET** | RW | ECC Bits Error Injection Enable.<br>This field enable ECC error injection to ECC bits part. There are four sets buffer in each port. Each bit controls one set buffer of the ports. Here control bits are both valid for port0 and port1.<br>0b - ECC bits error injection disable.<br>1b - ECC bits error injection enable. |
| **27-24**<br>**EIE_SET** | RW | Data Bits Error Injection Enable.<br>This field enable ECC error injection to data bits part. There are four sets buffer in each port. Each bit controls one set buffer of the ports. Here control bits are both valid for port0 and port1.<br>0b - Data bits error injection disable.<br>1b - Data bits error injection enable. |
| **23-20**<br>**EDATAC_POS2** | RW | ECC Error Position 2<br>Error bits position is specified by EDATAC_POS2 and EDATAC_POS1. If the values of EDATAC_POS2 and EDATAC_POS1 are different, then two error bits are injected. Otherwise, only one error bit is valid. |

| | | |
|---|---|---|
| **19-16**<br>**EDATAC_POS1** | RW | ECC Error Position 1<br>Error bits position is specified by EDATAC_POS2 and EDATAC_POS1. If the values of EDATAC_POS2 and EDATAC_POS1 are different, then two error bits are injected. Otherwise, only one error bit is valid. |
| **7**<br>**DDBEE** | RW | Disable double bits ECC Error Exception.<br>If this bit is set, system bus fault response will not be generated when a double-bit ECC error occurs.<br>0b - Error response is generated for a double-bit ECC error.<br>1b - Error response isn't generated for a double-bit ECC error. |
| **0**<br>**DES** | RW | Data Error Suppression<br>This bit could disable ECC events for DFLASH read access.<br>0b - ECC events of DFLASH access are reported.<br>1b - ECC events of DFLASH access are suppressed. |

### 28.6.5 FMC_FEIPC (Flash ECC Inject Position Control Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | EDATA_POS2 | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | EDATA_POS1 | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_FEIPC field descriptions**

| Field | Type | Description |
|---|---|---|
| **8-0**<br>**EDATA_POS1** | RW | Data Error Position 1<br>Error bits position is specified by EDATA_POS2 and EDATA_POS1. If the values of EDATA_POS2 and EDATA_POS1 are different, then two error bits are injected. Otherwise, only one error bit is valid. |
| **24-16**<br>**EDATA_POS2** | RW | Data Error Position 2<br>Error bits position is specified by EDATA_POS2 and EDATA_POS1. If the values of EDATA_POS2 and EDATA_POS1 are different, then two error bits are injected. Otherwise, only one error bit is valid. |

### 28.6.6 FMC_FPESA_L (Flash Program Erase Start Address Logical Register)

Offset : 300h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PESA_L | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PESA_L | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_FPESA_L field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**PESA_L** | RW | Program Erase Logical Start Address<br>Logical address for flash program or erase. |

### 28.6.7 FMC_FPESA_P (Flash Program Erase Start Address Physical Register)

Offset : 304h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PESA_P | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PESA_P | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_FPESA_P field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**PESA_P** | RO | Program Erase Physical Start Address<br>Flash physical address for flash program or erase. |

### 28.6.8 FMC_FB_FPELCKn (Flash Block n Fine Program Erase Lock Register)

The Block n Fine Program Erase Lock registers are used to protect flash fine from unexpected modification. Fine protection is only available for the last 256 KB of every block (for 256KB blocks, all sectors are available for protection).

Offset : 340h+N*4h (N : 0 ~ 4)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FPELCK | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FPELCK | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**FMC_FB_FPELCKn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**FPELCK** | RW | Fine Program Erase Lock<br>Each lock bit controls one Fine (8KB). If the lock bit is set to 1, the corresponding fine is locked. Program and Erase operation to locked fine could cause protection error.<br>Bit value 0 - Fine is available for program and erase operation.<br>Bit value 1 - Fine is locked for program and erase operations. |

### 28.6.9 FMC_FN_FPELCK (Flash NVR Fine Program Erase Lock Register)

Offset : 358h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | FPEL |
| W | | | | | | | | | | | | | | | | CK |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**FMC_FN_FPELCK field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**FPELCK** | RW | Fine Program Erase Lock<br>Each lock bit controls one fine region (40KB). If the lock bit is set to 1, the corresponding fine region is locked. Program and Erase operation to locked fine could cause protection error.<br>0b - Fine is available for program and erase operation.<br>1b - Fine is locked for program and erase operations. |

### 28.6.10 FMC_FB_CPELCKn (Flash Block n Coarse Program Erase Lock Register)

The Block n Coarse Program Erase Lock registers are used to protect flash fine from unexpected modification. Coarse protection is available for the remaining region of flash except for the last 256K.

Offset : 35Ch+N*4h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CPELCK | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CPELCK | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**FMC_FB_CPELCKn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**CPELCK** | RW | Coarse Program Erase Lock<br>Each lock bit controls one coarse region (64KB). If the lock bit is set to 1, the corresponding coarse region is locked. Program and Erase operation to locked fine could cause protection error.<br>Bit value 0 - Fine is available for program and erase operation.<br>Bit value 1 - Fine is locked for program and erase operations. |

### 28.6.11 FMC_OTA_CTRLn (FLASH OTA Control Register)

Offset : 500h+N*4h (N : 0 ~ 1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | | | | | | | | | OTA_LOCK | OTA_ACTIVE | OTA_EN | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_OTA_CTRLn field descriptions**

| Field | Type | Description |
|---|---|---|
| **6**<br>**OTA_LOCK** | RW | OTA Register Lock<br>0b - OTA_CTRL can be written.<br>1b - OTA_CTRL is locked. |
| **5**<br>**OTA_ACTIVE** | RW | OTA Active Block<br>0b - Flash block0/1 is the currently active OTA region.<br>1b - Flash block2/3 is the currently active OTA region. |
| **4-0**<br>**OTA_EN** | RW | OTA Enable<br>0xA means OTA enabled. Other values indicate OTA disabled. |

### 28.6.12 FMC_OTA_VER_LOCn (FLASH OTA Version Location Register)

Offset : 508h+N*4h (N : 0 ~ 1)

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

**FMC_OTA_VER_LOCn field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **19-0**<br>**OTA_VER_LOC** | RO | OTA Version Location<br>The location only covers the address in PFLASH. Software can read the OTA_VER_LOC address first to get flash block0/1/2/3 version location, then compare the corresponding block0/1 and block2/3 version value to decide which PFLASH should be set to ACTIVE. |

### 28.6.13 FMC_OTA_ACT_VERn (FLASH OTA Active Version Register)

Offset : 510h+N*4h (N : 0 ~ 1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OTA_ACT_VER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FMC_OTA_ACT_VERn field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**OTA_ACT_VER** | RO | OTA Active Version<br>This register indicates the currently active OTA version. |

# Chapter 29 ARMv7 Cache Controller (ACC)

## 29.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

This chapter describes the ARMv7 Cache Controller (ACC) in the following sections:

- Chip-specific Information
- Features
- Overview
- Functional Description
- Registers

## 29.2 Chip-specific Information

N/A

## 29.3 Features

The cache has the below features:

- ARMv7 L1 cache. For detailed information, refer to *ARM Cortex-M7 Processor Technical Reference Manual*.
- 16 KB Instruction Cache (ICACHE) and 8 KB Data Cache (DCACHE) for each core.
- DCACHE and ICACHE are disabled by default. Software can enable/disable the cache by configuring registers.
- Software can invalidate the entire ICACHE.
- Software can invalidate one instruction cache line by the Modified Virtual Address (MVA).
- Software can invalidate one data cache line by MVA.
- Software can invalidate DCACHE by set-way.
- Software can clean one data cache line by MVA.
- Software can clean DCACHE by set-way.
- TAG RAM and DATA RAM both support ECC check and ECC error report interrupt. For details, refer to the ERM chapter.
- TAG RAM and DATA RAM both support ECC error injection via the EIM module. For more information, refer to the EIM chapter.
- TAG RAM and DATA RAM ECC errors can be reported by the ERM module. For more information, refer to the ERM chapter.

## 29.4 Overview

A cache is a high-speed memory that can decrease the average time of a memory access via storing address and data information in its own memory when accessing the data in FLASH, RAM, and ROM for the first time.

## 29.5 Functional Description

### 29.5.1 DCACHE Enable

The DCACHE is disabled at reset and the cache tag and data array are not cleared at reset. Thus, software needs to clear the cache before enabling it. That is:

1) Configure DCISW register to invalidate the entire data cache by set-way.
2) Set CCR[DC] to enable DCACHE.

### 29.5.2 ICACEHE Enable

The ICACHE is disabled at reset and the cache tag and data array are not cleared at reset. Thus, software needs to clear the cache before enabling it. That is:

1) Invalidate the entire instruction cache by writing ICIALLU register with 0.
2) Set CCR[IC] to enable the ICACHE.

### 29.5.3 DCACHE Disable

Software needs to follow the steps to disable DCACHE:

1) Clear CCR[DC] to disable DCACHE.
2) Configure DCCISW register to clean and invalidate the entire DCACHE by set-way.

### 29.5.4 ICACHE Disable

Software needs to follow the below steps to disable ICACHE:

1) Clear CCR[IC] to disable ICACHE.
2) Invalidate the entire instruction cache by writing ICIALLU with 0.

## 29.6 Register Definition

ACC base address : E000_ED00h

**Register Overview - ACC**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **14h** | ACC_CCR (Configuration and Control Register) | 32 | RW | 0004_0200h | page |
| **78h** | ACC_CLIDR (Cache Level ID Register) | 32 | RO | 0900_0003h | page |
| **7Ch** | ACC_CTR (Cache Type Register) | 32 | RO | 8303_0003h | page |
| **80h** | ACC_CCSIDR (Cache Size ID Register) | 32 | RO | F007_E019h | page |
| **84h** | ACC_CSSELR (Cache Size Selection Register) | 32 | RW | 0000_0000h | page |
| **250h** | ACC_ICIALLU (I-cache Invalidate All to PoU Register) | 32 | WO | 0000_0000h | page |
| **258h** | ACC_ICIMVAU (I-cache Invalidate by MVA to PoU Register) | 32 | WO | 0000_0000h | page |
| **25Ch** | ACC_DCIMVAC (D-cache Invalidate by MVA to PoC Register) | 32 | WO | 0000_0000h | page |
| **260h** | ACC_DCISW (D-cache Invalidate by Set-way Register) | 32 | WO | 0000_0000h | page |
| **264h** | ACC_DCCMVAU (D-cache Clean by MVA to PoU Register) | 32 | WO | 0000_0000h | page |
| **268h** | ACC_DCCMVAC (D-cache Clean by MVA to PoC Register) | 32 | WO | 0000_0000h | page |
| **26Ch** | ACC_DCCSW (D-cache Clean by Set-way Register) | 32 | WO | 0000_0000h | page |
| **270h** | ACC_DCCIMVAC (D-cache Clean and Invalidate by MVA to PoC Register) | 32 | WO | 0000_0000h | page |
| **274h** | ACC_DCCISW (D-cache Clean and Invalidate by Set-way Register) | 32 | WO | 0000_0000h | page |
| **29Ch** | ACC_CACR (L1 Cache Control Register) | 32 | RW | 0000_0000h | page |
| **2B0h+n*4h** | ACC_IEBRn (Instruction Error Bank Register) | 32 | RW | 0000_0000h | page |
| **2B8h+n*4h** | ACC_DEBRn (Data Error Bank Register) | 32 | RW | 0000_0000h | page |

### 29.6.1 ACC_CCR (Configuration and Control Register)

Sets or returns configuration and control data, and provides control over caching and branch prediction.

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | BP | IC | DC |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | STKALIGN | BFHFNMIGN | 0 | | | DIV_0_TRP | UNALIGN_TRP | 0 | USERSETMPEND | NONBASETHRDENA |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ACC_CCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **18**<br>**BP** | RW | Branch Prediction Enable<br>0b - Program flow prediction disabled.<br>1b - Program flow prediction enabled.<br>Setting this bit to 1 enables branch prediction, also called program flow prediction. If program flow prediction cannot be disabled, this bit is RAO/WI. If the implementation does not support program flow prediction, this bit is RAZ/WI. |
| **17**<br>**IC** | RW | Instruction Cache Enable<br>This is a global enable bit for instruction caches.<br>0b - Instruction caches disabled.<br>1b - Instruction caches enabled.<br>If the system does not implement any instruction caches that can be accessed by the processor at any level of the memory hierarchy, this bit is RAZ/WI. If the system implements any instruction caches that can be accessed by the processor then it must be possible to disable them by setting this bit to 0. |

| | | |
|---|---|---|
| **16**<br>**DC** | RW | Data Cache Enable<br>This is a global enable bit for data and unified caches.<br>0b - Data and unified caches disabled.<br>1b - Data and unified caches enabled.<br>If the system does not implement any data or unified caches that can be accessed by the processor at any level of the memory hierarchy, this bit is RAZ/WI. If the system implements any data or unified caches that can be accessed by the processor then it must be possible to disable them by setting this bit to 0. |
| **9**<br>**STKALIGN** | RW | Determines whether the exception entry sequence guarantees 8-byte stack frame alignment,<br>adjusting the SP if necessary before saving state.<br>0b - Guaranteed SP alignment is 4-byte, no SP adjustment is performed.<br>1b - 8-byte alignment guaranteed, SP adjusted if necessary. |
| **8**<br>**BFHFNMIGN** | RW | Determines the effect of precise data access faults on handlers running at priority -1 or<br>priority -2.<br>0b - Precise data access fault causes a lockup.<br>1b - Handler ignores the fault. |
| **4**<br>**DIV_0_TRP** | RW | Controls the trap on divide by 0.<br>0b - Trapping disabled.<br>1b - Trapping enabled. |
| **3**<br>**UNALIGN_TRP** | RW | Controls the trapping of unaligned word or halfword accesses.<br>0b - Trapping disabled.<br>1b - Trapping enabled.<br>Unaligned load-store multiples and word or halfword exclusive accesses always fault. |
| **1**<br>**USERSETMPEND** | RW | Controls whether unprivileged software can access the STIR(Software Triggered Interrupt Register).<br>0b - Unprivileged software cannot access the STIR.<br>1b - Unprivileged software can access the STIR. |
| **0**<br>**NONBASETHRDENA** | RW | Controls whether the processor can enter Thread mode with exceptions active.<br>0b - Any attempt to return to Thread mode will result in an exception if the number of active exceptions is non-zero and does not rely on execution priority boosting including BASEPRI, FAULTMASK and PRIMASK.<br>1b - The processor can enter Thread mode with exceptions active because of a controlled return value. |

### 29.6.2 ACC_CLIDR (Cache Level ID Register)

CLIDR is a read-only register, can be accessed in privilege mode only. Indicates the cache levels that are implemented. Architecturally, there can be a different number of cache levels on the instruction and data side.

Offset : 78h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | LOUU | | | LOC | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | LOCT | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**ACC_CLIDR field descriptions**

| Field | Type | Description |
|---|---|---|
| **29-27**<br>**LOUU** | RO | Level of Unification Uniprocessor<br>000b - Level 1, if neither instruction nor data cache is implemented.<br>001b - Level 2, if either cache is implemented. |
| **26-24**<br>**LOC** | RO | Level of Coherency<br>000b - Level 1, if neither instruction nor data cache is implemented.<br>001b - Level 2, if either cache is implemented. |
| **2-0**<br>**LOCT** | RO | Level 1 Cache Type<br>000b - No instruction or data cache is implemented.<br>001b - Instruction cache is implemented.<br>010b - Data cache is implemented.<br>011b - Instruction and data cache are implemented. |

### 29.6.3 ACC_CTR (Cache Type Register)

The CTR provides information about the architecture of the caches. Only accessible by Privileged software.

Offset : 7Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FMT | | | 0 | CWG | | | | ERG | | | | DMINLINE | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | IMINLINE | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**ACC_CTR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-29**<br>**FMT** | RO | Indicates the implemented CTR format. The possible values of this are:<br>000b - No caches are implemented and no cache type information is provided. All remaining fields in this register are RAZ including fields that are marked as RAO.<br>100b - ARMv7 format. |
| **27-24**<br>**CWG** | RO | Cache Write-back Granule<br>The maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified, encoded as Log2 of the number of words. A value of 0000b indicates that the CTR does not provide cache write-back Granule information and either:<br>• The architectural maximum of 512 words (2Kbytes) must be assumed.<br>• The Cache write-back Granule can be determined from maximum cache line size encoded in the Cache Size ID Registers.<br>Values greater than 1001b are reserved. |
| **23-20**<br>**ERG** | RO | Exclusives Reservation Granule<br>The maximum size of the reservation granule that has been implemented for the Load-Exclusive and Store-Exclusive instructions, encoded as Log2 of the number of words. A value of 0000b indicates that the CTR does not provide Exclusives Reservation Granule information and the architectural maximum of 512 words (2Kbytes) must be assumed.<br>Values greater than 1001b are reserved. |
| **19-16**<br>**DMINLINE** | RO | Log2 of the number of words in the smallest cache line of all the data caches and unified caches that are controlled by the processor. |
| **3-0**<br>**IMINLINE** | RO | Log2 of the number of words in the smallest cache line of all the instruction caches that are controlled by the processor. |

### 29.6.4 ACC_CCSIDR (Cache Size ID Register)

Only accessible by Privileged software. If CSSELR indicates a cache that is not implemented, the result of reading CCSIDR is UNPREDICTABLE. The implementation includes one CCSIDR for each cache that it can access. CSSELR selects which Cache Size ID register is accessible.

Offset : 80h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WT | WB | RA | WA | NUMSETS | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NUMSETS | | | ASSOCIATIVITY | | | | | | | | | | LINESIZE | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

**ACC_CCSIDR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**WT** | RO | Indicates whether the cache level supports Write-Through. |
| **30**<br>**WB** | RO | Indicates whether the cache level supports write-back. |
| **29**<br>**RA** | RO | Indicates whether the cache level supports read-allocation. |
| **28**<br>**WA** | RO | Indicates whether the cache level supports write-allocation. |
| **27-13**<br>**NUMSETS** | RO | Number of sets in cache -1<br>Therefore a value of 0 indicates 1 set in the cache. The number of sets does not have to be a power of 2. |
| **12-3**<br>**ASSOCIATIVITY** | RO | Associativity of cache -1<br>Therefore a value of 0 indicates an associativity of 1. The associativity does not have to be a power of 2. |

| | | |
|---|---|---|
| **2-0**<br>**LINESIZE** | RO | Log2(Number of words in cache line) -2<br>For example:<br>• For a line length of 4 words: Log2(4) = 2, LineSize entry = 0. This is the minimum line length.<br>• For a line length of 8 words: Log2(8) = 3, LineSize entry = 1. |

### 29.6.5 ACC_CSSELR (Cache Size Selection Register)

The CSSELR selects the current CCSIDR, by specifying:

- The required cache level.
- The cache type, either:
  - Instruction cache, if the memory system implements separate instruction and data caches.
  - Data cache. The data cache argument must be used for a unified cache.

Offset : 84h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | LEVEL | | | IND |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ACC_CSSELR field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-1**<br>**LEVEL** | RW | Cache Level of Required Cache<br>Permitted values are from 000b, indicating Level 1 cache, to 110b indicating Level 7 cache. |
| **0**<br>**IND** | RW | Instruction not Data Bit<br>Permitted values are:<br>0b - Data or unified cache.<br>1b - Instruction cache. |

### 29.6.6 ACC_ICIALLU (I-cache Invalidate All to PoU Register)

Offset : 250h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | ICIALLU | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | ICIALLU | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ACC_ICIALLU field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**ICIALLU** | WO | I-cache invalidate all.<br>Writing 0 to ICIALLU register can invalidate the entire instrunction cache. |

### 29.6.7 ACC_ICIMVAU (I-cache Invalidate by MVA to PoU Register)

This instruction invalidates an instruction cache line based on the address it contains.

Offset : 258h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | MVA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | 0 | | |
| W | MVA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ACC_ICIMVAU field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-5**<br>**MVA** | WO | MVA of Requested Operation |

### 29.6.8 ACC_DCIMVAC (D-cache Invalidate by MVA to PoC Register)

This instruction invalidates an data cache line based on the address.

Offset : 25Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | MVA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | 0 | | | | |
| W | MVA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ACC_DCIMVAC field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-5**<br>**MVA** | WO | MVA of requested operation |

### 29.6.9 ACC_DCISW (D-cache Invalidate by Set-way Register)

Offset : 260h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | 0 | | | | | | | | | | | | | |
| W | WAY | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | 0 | | | | | | | | | | | 0 | | |
| W | | | SET | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ACC_DCISW field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-30**<br>**WAY** | WO | Way that operation applies to.<br>For the data cache, values 0, 1, 2 and 3 are supported. |

| | | |
|---|---|---|
| **13-5 SET** | WO | Set/index that operation applies to. The number of indices in a cache depends on the configured cache size. When this is less than the maximum, use the LSB of this field. The number of sets in the cache can be determined by reading the Cache Size ID Register. |

### 29.6.10 ACC_DCCMVAU (D-cache Clean by MVA to PoU Register)

This instruction cleans an data cache line based on the address.

Offset : 264h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | MVA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | 0 | | | | |
| W | MVA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ACC_DCCMVAU field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-5 MVA** | WO | MVA of requested operation |

### 29.6.11 ACC_DCCMVAC (D-cache Clean by MVA to PoC Register)

This instruction cleans an data cache line based on the address.

Offset : 268h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | MVA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | 0 | | | | |
| W | MVA | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ACC_DCCMVAC field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-5**<br>**MVA** | WO | MVA of requested operation |

### 29.6.12 ACC_DCCSW (D-cache Clean by Set-way Register)

Offset : 26Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | 0 | | | | | | | | | | | | | |
| W | WAY | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | 0 | | | | | | | | | 0 | | | | |
| W | | | SET | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ACC_DCCSW field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-30**<br>**WAY** | WO | Way that operation applies to.<br>For the data cache, values 0, 1, 2 and 3 are supported. |
| **13-5**<br>**SET** | WO | Set/index that operation applies to. The number of indices in a cache depends on the configured cache size. When this is less than the maximum, use the LSB of this field. The number of sets in the cache can be determined by reading the Cache Size ID Register. |

### 29.6.13 ACC_DCCIMVAC (D-cache Clean and Invalidate by MVA to PoC Register)

This instruction cleans and invalidates an data cache line based on the address.

Offset : 270h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | MVA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | 0 | | | | |
| W | MVA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ACC_DCCIMVAC field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-5**<br>**MVA** | WO | MVA of requested operation |

### 29.6.14 ACC_DCCISW (D-cache Clean and Invalidate by Set-way Register)

Offset : 274h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | 0 | | | | | | | | | | | | | |
| W | WAY | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | 0 | | | | | | | | | | | 0 | | |
| W | | | SET | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ACC_DCCISW field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-30**<br>**WAY** | WO | Way that operation applies to.<br>For the data cache, values 0, 1, 2 and 3 are supported. |
| **13-5**<br>**SET** | WO | Set/index that operation applies to. The number of indices in a cache depends on the configured cache size. When this is less than the maximum, use the LSB of this field. The number of sets in the cache can be determined by reading the Cache Size ID Register. |

### 29.6.15 ACC_CACR (L1 Cache Control Register)

The CACR controls the optional L1 ECC and the L1 cache coherency usage model.

Offset : 29Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | FORCEWT | ECCDIS | SIWT |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ACC_CACR field descriptions**

| Field | Type | Description |
|---|---|---|
| **2**<br>**FORCEWT** | RW | Enables Force Write-Through in the data cache:<br>0b - Disables Force Write-Through.<br>1b - Enables Force Write-Through. All Cacheable memory regions are treated as Write-Through.<br>This bit is RAZ/WI if the data cache is excluded. If the data cache is included the reset value of FORCEWT is 0. |
| **1**<br>**ECCDIS** | RW | Enables ECC in the instruction and data cache:<br>0b - Enables ECC in the instruction and data cache. This is RAO/WI if both data cache and instruction cache are excluded or if ECC is excluded.<br>1b - Disables ECC in the instruction and data cache. This is RAZ/WI if data cache is not configured. |
| **0**<br>**SIWT** | RW | Shared cacheable-is-WT for data cache. Enables cache coherency usage:<br>0b - Normal Cacheable Shareable locations are treated as being Non-cacheable. Programmed inner cacheability attributes are ignored. This is the default mode of operation for shared memory. The data cache is transparent to software for these locations and therefore no software maintenance is required to maintain coherency.<br>1b - Normal Cacheable Shareable locations are treated as Write-Through. Programmed inner cacheability attributes are ignored. All writes are globally visible. Other memory agent updates are not visible to Cortex-M7 processor software without suitable cache maintenance. Useful for heterogeneous MP-systems where, for example, the Cortex-M7 processor is integrated on the Accelerator Coherency Port (ACP) interface on an MP-capable processor. |

### 29.6.16 ACC_IEBRn (Instruction Error Bank Register)

Stores information about the error detected in the instruction cache during a cache lookup.

Offset : 2B0h+N*4h (N : 0 ~ 1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | TOE | RB |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RL | | | | | | | | | | | | | | LOCKED | VALID |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ACC_IEBRn field descriptions**

| Field | Type | Description |
|---|---|---|
| **17**<br>**TOE** | RW | Indicates the error type:<br>0b - Correctable error.<br>1b - Non-correctable error. |
| **16**<br>**RB** | RW | Indicates which RAM bank to use:<br>0b - Tag RAM.<br>1b - Data RAM. |
| **15-2**<br>**RL** | RW | Indicates the location in instruction cache RAM:<br>[14] Way.<br>[13:4] Index.<br>[3:2] Line doubleword offset. |
| **1**<br>**LOCKED** | RW | Indicates whether the location is locked or not locked:<br>0b - Location is not locked and available for hardware to allocate.<br>1b - Location is locked by software. Hardware is not allowed to allocate to this entry.<br>Reset by powerup reset to 0. |
| **0**<br>**VALID** | RW | Indicates whether the entry is valid or not:<br>0b - Entry is invalid.<br>1b - Entry is valid.<br>Reset by powerup reset to 0. |

### 29.6.17 ACC_DEBRn (Data Error Bank Register)

Stores information about the error detected in the data cache during a cache lookup.

Offset : 2B8h+N*4h (N : 0 ~ 1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | TOE | RB |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RL | | | | | | | | | | | | | | LOCKED | VALID |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ACC_DEBRn field descriptions**

| Field | Type | Description |
|---|---|---|
| **17**<br>**TOE** | RW | Indicates the error type:<br>0b - Correctable error.<br>1b - Non-correctable error. |
| **16**<br>**RB** | RW | Indicates which RAM bank to use:<br>0b - Tag RAM.<br>1b - Data RAM. |
| **15-2**<br>**RL** | RW | Indicates the location in data cache RAM:<br>[15:14] Way.<br>[13:4] Index.<br>[3:2] Line doubleword offset. |
| **1**<br>**LOCKED** | RW | Indicates whether the location is locked or not locked:<br>0b - Location is not locked and available for hardware to allocate.<br>1b - Location is locked by software. Hardware is not allowed to allocate to this entry.<br>Reset by powerup reset to 0. |
| **0**<br>**VALID** | RW | Indicates whether the entry is valid or not:<br>0b - Entry is invalid.<br>1b - Entry is valid.<br>Reset by powerup reset to 0. |

# Chapter 30 AHB Overlay

## 30.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • Updated the bit field descriptions in the register sections *AHB_OVERLAY_REGION0_MASK, AHB_OVERLAY_REGION1_MASK,* and *AHB_OVERLAY_REGION2_MASK;*<br>• Editorial changes |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

This chapter describes the AHB Overlay in the following sections:

- Chip-specific Information
- Features
- Overview
- Functional Description
- Registers

## 30.2 Chip-specific Information

There is one AHB Overlay module between the CM7 core and Flash. The module is used to achieve the address remapping. The clock frequency of the module is the bus clock frequency with the default value of 120 MHz.

## 30.3 Features

The overlay has the following features:

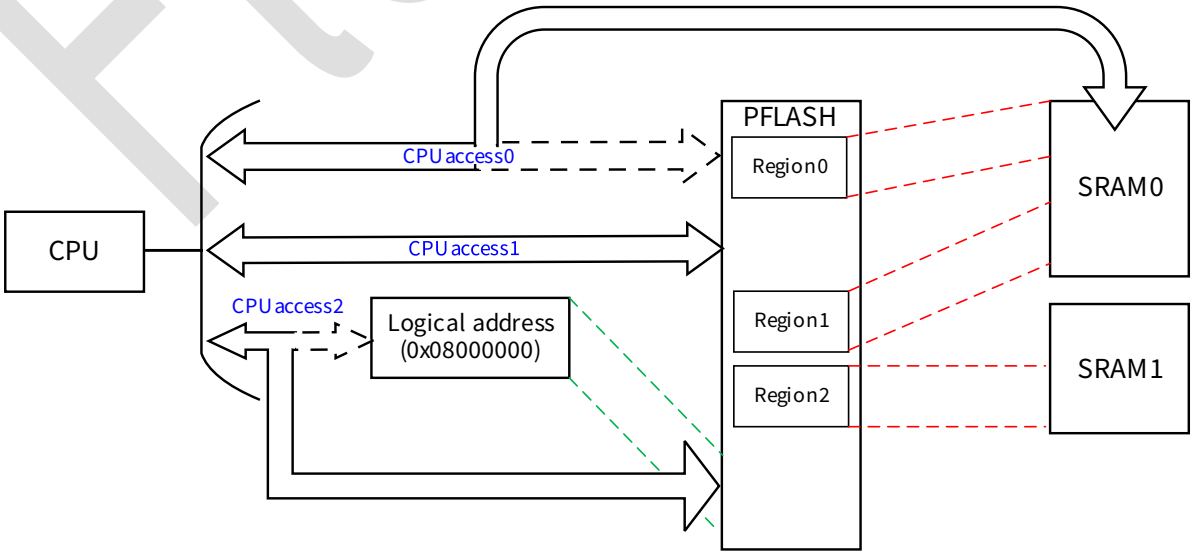
- Supported 3 regions for remapping PFlash addresses
- Each remapping region size is configurable, and the maximum size is 64 KB
- The destination of each region is configurable
- Overlapping between multiple regions should not be allowed, otherwise, an error interrupt will occur
- One extra region Flash Alias Region (FAR) support from logical address to PFlash remapping
- Fixed logical region (FAR) base address, 0x08000000
- The FAR remapping’ s destination address must be PFlash, otherwise, an error interrupt will occur
- The minimum size of the FAR region is 64KB, and the size can be changed by reconfiguring the register
- Each remapping region has an enable signal to control the region is enable or disable
- The FAR region also has an enable signal to control the region is enable or disable

## 30.4 Overview

The AHB Overlay acts as an address remapping controller. Any access that hits the three remapping regions will be remapped to the configured destination of each region. Any access that hits the FAR region will be remapped to a PFlash address.

A simplified block diagram of the overlay function is shown below.

**Figure 43. Overlay block diagram**



## 30.5 Functional Description

### 30.5.1 AHB Overlay Remapping Region

There are three regions for achieving address remapping. Each region has an independent enable signal to control the enable of a region. There is also a global enable signal to control all regions.

The source address of each region must be a PFlash address, the other base address will not be able to be configured into the register. The destination address of each region can be configured by software. The region size is also configurable, but the size only supports 1 KB, 2 KB, 4 KB, 8 KB, 16 KB, 32 KB, and 64KB; other sizes are invalid. If an invalid size is configured into the register, an error interrupt will occur. The source address and destination address must be aligned with the region size, otherwise, an error interrupt will occur.

Usually, the destination of each region is an SRAM base address, as shown in the overlay block diagram. The read PFlash accesses will be remapped to SRAM. The time for reading operations will be significantly reduced.

The regions cannot overlap. If an overlap occurs, an error interrupt flag will be set.

If one of the above errors occurs, the enable signal for the corresponding region cannot be configured and enabled until the error is corrected.

### 30.5.2 FAR Region

This extra FAR region is used to achieve remapping from a logical address to a PFlash address.

The source address of the FAR region is fixed at 0x08000000. The destination address is configurable, but the destination address must be configured to the PFlash address, otherwise, an error interrupt will occur.

The size of the FAR region is also configurable by writing the register FAR_SIZE via software. The FAR region size is (FAR_SIZE + 1) x 64KB. After FAR remapping, the final address of this mapped area cannot exceed the final address of PFlash, otherwise, an error interrupt will occur.

If one of the above errors occurs, the enable signal for the corresponding region cannot be configured and enabled until the error is corrected.

## 30.6 Register Definition

AHB_overlay base address : E008_1000h

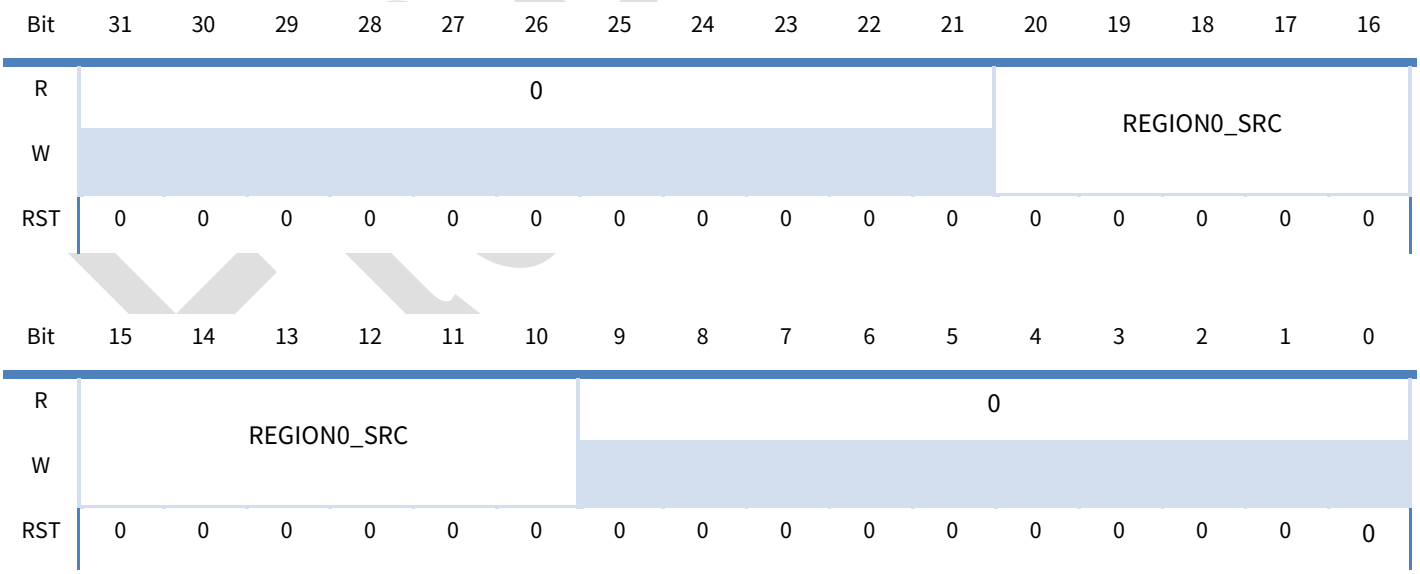
**Register Overview - AHB_OVERLAY**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **00h** | AHB_OVERLAY_REGION_0_SRC (Region 0 Source Address Register) | 32 | RW | 0000_0000h | page |
| **04h** | AHB_OVERLAY_REGION_0_DST (Region 0 Destination Address Register) | 32 | RW | 0000_0000h | page |
| **08h** | AHB_OVERLAY_REGION_0_SIZE (Region 0 Size Register) | 32 | RW | 0000_0001h | page |
| **0Ch** | AHB_OVERLAY_REGION_0_EN (Region 0 Enable Register) | 32 | RW | 0000_0000h | page |
| **10h** | AHB_OVERLAY_REGION_1_SRC (Region 1 Source Address Register) | 32 | RW | 0000_0000h | page |
| **14h** | AHB_OVERLAY_REGION_1_DST (Region 1 Destination Address Register) | 32 | RW | 0000_0000h | page |
| **18h** | AHB_OVERLAY_REGION_1_SIZE (Region 1 Size Register) | 32 | RW | 0000_0001h | page |

| | | | | | |
|---|---|---|---|---|---|
| **1Ch** | AHB_OVERLAY_REGION_1_EN (Region 1 Enable Register) | 32 | RW | 0000_0000h | page |
| **20h** | AHB_OVERLAY_REGION_2_SRC (Region 2 Source Address Register) | 32 | RW | 0000_0000h | page |
| **24h** | AHB_OVERLAY_REGION_2_DST (Region 2 Destination Address Register) | 32 | RW | 0000_0000h | page |
| **28h** | AHB_OVERLAY_REGION_2_SIZE (Region 2 Size Register) | 32 | RW | 0000_0001h | page |
| **2Ch** | AHB_OVERLAY_REGION_2_EN (Region 2 Enable Register) | 32 | RW | 0000_0000h | page |
| **80h** | AHB_OVERLAY_GLOBAL_EN (FAR and Overlay Enable Register) | 32 | RW | 0000_0002h | page |
| **84h** | AHB_OVERLAY_INTR_FLAG (Interrupt Flag Register) | 32 | RO | 0000_0000h | page |
| **88h** | AHB_OVERLAY_INTR_CLR (Interrupt Clear Register) | 32 | WO | 0000_0000h | page |
| **8Ch** | AHB_OVERLAY_INTR_EN (Interrupt Enable Register) | 32 | RW | 0000_0000h | page |
| **90h** | AHB_OVERLAY_FAR_SRC (FAR Source Register) | 32 | RO | 0800_0000h | page |
| **94h** | AHB_OVERLAY_FAR_DST (FAR Destination Register) | 32 | RW | 0100_0000h | page |
| **98h** | AHB_OVERLAY_FAR_SIZE (FAR Size Register) | 32 | RW | 0000_0000h | page |
| **A0h** | AHB_OVERLAY_REGION0_MASK (Region 0 Mask Register) | 32 | RO | FFFF_FC00h | page |
| **A4h** | AHB_OVERLAY_REGION1_MASK (Region 1 Mask Register) | 32 | RO | FFFF_FC00h | page |
| **A8h** | AHB_OVERLAY_REGION2_MASK (Region 2 Mask Register) | 32 | RO | FFFF_FC00h | page |

### 30.6.1 AHB_OVERLAY_REGION_0_SRC (Region 0 Source Address Register)

Offset : 00h




**AHB_OVERLAY_REGION_0_SRC field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **20-10**<br>**REGION0_SRC** | RW | Region 0 Source Start Address<br>[9:0] is fixed 10'b0;<br>31:21 is fixed flash base address |

### 30.6.2 AHB_OVERLAY_REGION_0_DST (Region 0 Destination Address Register)

Offset : 04h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | REGION0_DST | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | REGION0_DST | | | | | | 0 | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_REGION_0_DST field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-10**<br>**REGION0_DST** | RW | Region 0 Destination Start Address<br>9:0 is fixed 10'b0 |

### 30.6.3 AHB_OVERLAY_REGION_0_SIZE (Region 0 Size Register)

Offset : 08h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | | | | | | | | REGION0_SIZE | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**AHB_OVERLAY_REGION_0_SIZE field descriptions**

| Field | Type | Description |
|---|---|---|
| **6-0**<br>**REGION0_SIZE** | RW | Region 0 Size<br>Only supports:<br>7'h01: 1KB;<br>7'h02: 2KB;<br>7'h04: 4KB;<br>7'h08: 8KB;<br>7'h10: 16KB;<br>7'h20: 32KB;<br>7'h40: 64KB;<br>Other values are invalid. |

### 30.6.4 AHB_OVERLAY_REGION_0_EN (Region 0 Enable Register)

Offset : 0Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | REGION0_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_REGION_0_EN field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**REGION0_EN** | RW | Enable of Region 0 |

### 30.6.5 AHB_OVERLAY_REGION_1_SRC (Region 1 Source Address Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | REGION1_SRC | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REGION1_SRC | | | | | | | | | | 0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_REGION_1_SRC field descriptions**

| Field | Type | Description |
|---|---|---|
| **20-10**<br>**REGION1_SRC** | RW | Region 1 Source Start Address<br>[9:0] is fixed 10'b0;<br>31:21 is fixed flash base address |

### 30.6.6 AHB_OVERLAY_REGION_1_DST (Region 1 Destination Address Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REGION1_DST | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REGION1_DST | | | | | | | | | | 0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_REGION_1_DST field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-10**<br>**REGION1_DST** | RW | Region 1 Destination Start Address<br>9:0 is fixed 10'b0; |

### 30.6.7 AHB_OVERLAY_REGION_1_SIZE (Region 1 Size Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | REGION1_SIZE | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**AHB_OVERLAY_REGION_1_SIZE field descriptions**

| Field | Type | Description |
|---|---|---|
| **6-0**<br>**REGION1_SIZE** | RW | Region 1 Size<br>Only supports:<br>7'h01: 1KB;<br>7'h02: 2KB;<br>7'h04: 4KB;<br>7'h08: 8KB;<br>7'h10: 16KB;<br>7'h20: 32KB;<br>7'h40: 64KB;<br>Other values are invalid. |

### 30.6.8 AHB_OVERLAY_REGION_1_EN (Region 1 Enable Register)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | REGION1_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_REGION_1_EN field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **0**<br>**REGION1_EN** | RW | Enable of Region 1 |

### 30.6.9 AHB_OVERLAY_REGION_2_SRC (Region 2 Source Address Register)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | REGION2_SRC | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REGION2_SRC | | | | | | 0 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_REGION_2_SRC field descriptions**

| Field | Type | Description |
|---|---|---|
| **20-10**<br>**REGION2_SRC** | RW | Region 2 Source Start Address<br>[9:0] is fixed 10'b0;<br>31:21 is fixed flash base address |

### 30.6.10 AHB_OVERLAY_REGION_2_DST (Region 2 Destination Address Register)

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REGION2_DST | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REGION2_DST | | | | | | 0 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_REGION_2_DST field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-10**<br>**REGION2_DST** | RW | Region 2 Destination Start Address<br>9:0 is fixed 10'b0 |

### 30.6.11 AHB_OVERLAY_REGION_2_SIZE (Region 2 Size Register)

Offset : 28h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | REGION2_SIZE | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**AHB_OVERLAY_REGION_2_SIZE field descriptions**

| Field | Type | Description |
|---|---|---|
| **6-0**<br>**REGION2_SIZE** | RW | Region 2 Size<br>Only supports:<br>7'h01: 1KB;<br>7'h02: 2KB;<br>7'h04: 4KB;<br>7'h08: 8KB;<br>7'h10: 16KB;<br>7'h20: 32KB;<br>7'h40: 64KB;<br>Other values are invalid. |

### 30.6.12 AHB_OVERLAY_REGION_2_EN (Region 2 Enable Register)

Offset : 2Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | REGION2_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_REGION_2_EN field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**REGION2_EN** | RW | Enable of Region 2 |

### 30.6.13 AHB_OVERLAY_GLOBAL_EN (FAR and Overlay Enable Register)

Offset : 80h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | FAR_EN | OVERLAY_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

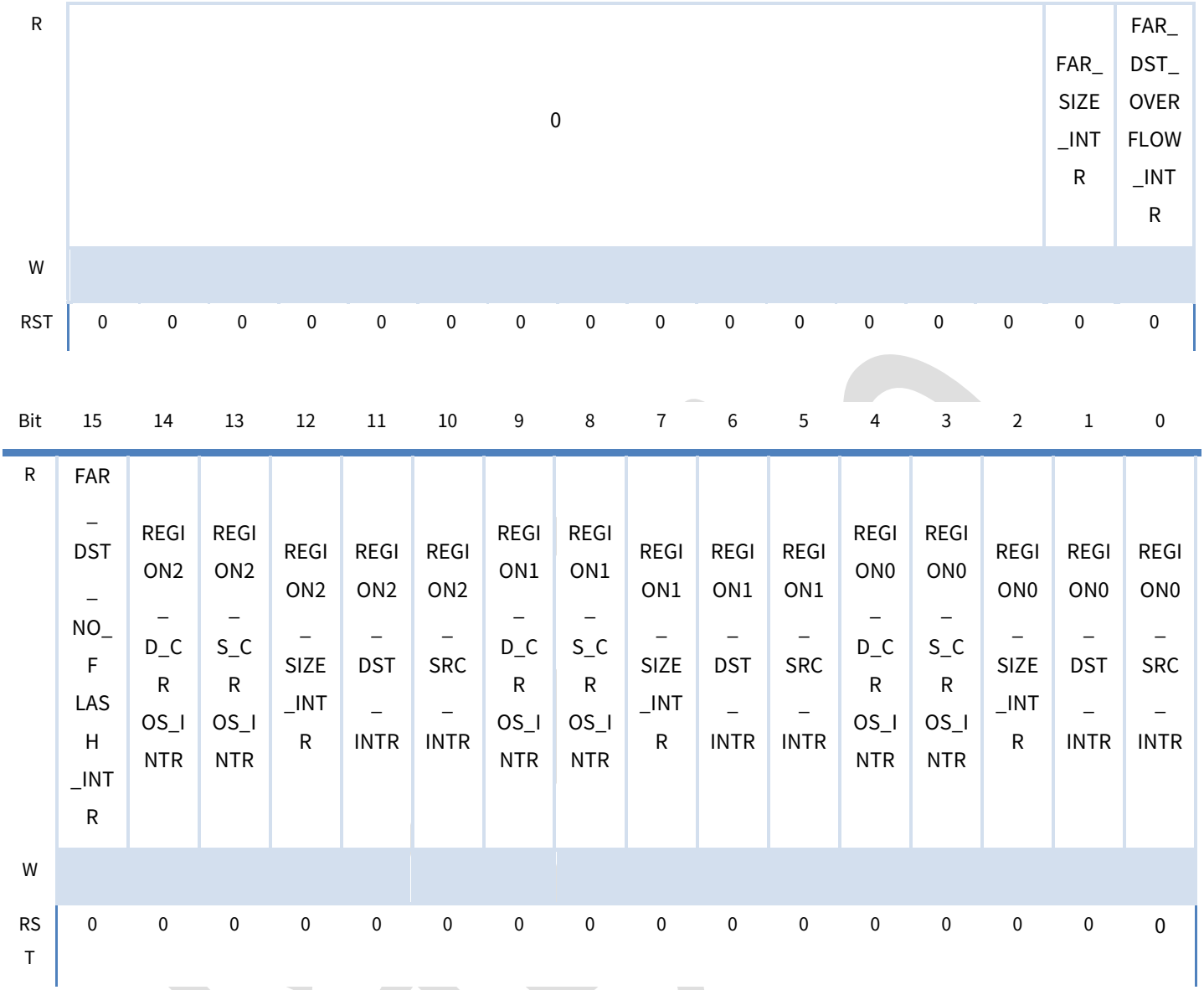
**AHB_OVERLAY_GLOBAL_EN field descriptions**

| Field | Type | Description |
|---|---|---|
| **1**<br>**FAR_EN** | RW | Enable of FAR |
| **0**<br>**OVERLAY_EN** | RW | Enable of Overlay |

### 30.6.14 AHB_OVERLAY_INTR_FLAG (Interrupt Flag Register)

Offset : 84h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|



| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | FAR_SIZE_INTR | FAR_DST_OVERFLOW_INTR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FAR_DST_NO_FLASH_INTR | REGION2_D_CROS_INTR | REGION2_S_CROS_INTR | REGION2_SIZE_INTR | REGION2_DST_INTR | REGION2_SRC_INTR | REGION1_D_CROS_INTR | REGION1_S_CROS_INTR | REGION1_SIZE_INTR | REGION1_DST_INTR | REGION1_SRC_INTR | REGION0_D_CROS_INTR | REGION0_S_CROS_INTR | REGION0_SIZE_INTR | REGION0_DST_INTR | REGION0_SRC_INTR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_INTR_FLAG field descriptions**

| Field | Type | Description |
|---|---|---|
| **17**<br>**FAR_SIZE_INTR** | RO | FAR Size Interrupt<br>0b - Invalid<br>1b - The FAR size value is invalid |
| **16**<br>**FAR_DST_OVERFLOW_INTR** | RO | FAR Destination Overflow Interrupt<br>0b - Invalid<br>1b - The FAR destination region is overflow with size |
| **15**<br>**FAR_DST_NO_FLASH_INTR** | RO | FAR Destination No Flash Interrupt<br>0b - Invalid<br>1b - The FAR destination address is not flash |

| | | |
|---|---|---|
| **14**<br>**REGION2_D_CROS_INTR** | RO | Region 2 Destination Cross Interrupt<br>0b - Invalid<br>1b - The range of region 2 destination address is crossed with other region |
| **13**<br>**REGION2_S_CROS_INTR** | RO | Region 2 Source Cross Interrupt<br>0b - Invalid<br>1b - The range of region 2 source address is crossed with other region |
| **12**<br>**REGION2_SIZE_INTR** | RO | Region 2 Size Interrupt<br>0b - Invalid<br>1b - The region 2 size value is invalid |
| **11**<br>**REGION2_DST_INTR** | RO | Region 2 Destination Interrupt<br>0b - Invalid<br>1b - The region 2 destination address is not aligned with size |
| **10**<br>**REGION2_SRC_INTR** | RO | Region 2 Source Interrupt<br>0b - Invalid<br>1b - The region 2 source address is not aligned with size |
| **9**<br>**REGION1_D_CROS_INTR** | RO | Region 1 Destination Cross Interrupt<br>0b - Invalid<br>1b - The range of region 1 destination address is crossed with other region |
| **8**<br>**REGION1_S_CROS_INTR** | RO | Region 1 Source Address Cross Interrupt<br>0b - Invalid<br>1b - The range of region 1 source address is crossed with other region |
| **7**<br>**REGION1_SIZE_INTR** | RO | Region 1 Size Interrupt<br>0b - Invalid<br>1b - The region 1 size value is invalid |
| **6**<br>**REGION1_DST_INTR** | RO | Region 1 Destination Interrupt<br>0b - Invalid<br>1b - The region 1 destination address is not aligned with size |
| **5**<br>**REGION1_SRC_INTR** | RO | Region 1 Source Address Interrupt<br>0b - Invalid<br>1b - The region 1 source address is not aligned with size |
| **4**<br>**REGION0_D_CROS_INTR** | RO | Region 0 Destination Cross Interrupt<br>0b - Invalid<br>1b - The range of region 0 destination address is crossed with other region |

| | | |
|---|---|---|
| **3**<br>**REGION0_S_CROS_INTR** | RO | Region 0 Source Cross Interrupt<br>0b - Invalid<br>1b - The range of region 0 source address is crossed with other region |
| **2**<br>**REGION0_SIZE_INTR** | RO | Region 0 Size Interrupt<br>0b - Invalid<br>1b - The region 0 size value is invalid |
| **1**<br>**REGION0_DST_INTR** | RO | Region 0 Destination Interrupt<br>0b - Invalid<br>1b - The region 0 destination address is not aligned with size |
| **0**<br>**REGION0_SRC_INTR** | RO | Region 0 Source Interrupt<br>0b - Invalid<br>1b - The region 0 source address is not aligned with size |

### 30.6.15 AHB_OVERLAY_INTR_CLR (Interupt Clear Register)

Offset : 88h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | FAR_SIZE_INTR_CLR | FAR_DST_OVERFLOW_INTR_CLR |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | FAR_DST_ON_FLASH_INTR_CLR | REGION2_D_CROS_INTR_CLR | REGION2_S_CROS_INTR_CLR | REGION2_SIZE_INTR_CLR | REGION2_DST_INTR_CLR | REGION2_SRC_INTR_CLR | REGION1_D_CROS_INTR_CLR | REGION1_S_CROS_INTR_CLR | REGION1_SIZE_INTR_CLR | REGION1_DST_INTR_CLR | REGION1_SRC_INTR_CLR | REGION0_D_CROS_INTR_CLR | REGION0_S_CROS_INTR_CLR | REGION0_SIZE_INTR_CLR | REGION0_DST_INTR_CLR | REGION0_SRC_INTR_CLR |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_INTR_CLR field descriptions**

| Field | Type | Description |
|---|---|---|
| **17**<br>**FAR_SIZE_INTR_CLR** | WO | FAR Size Interrupt Clear<br>Writing 1 to clear the FAR size interrupt. |
| **16**<br>**FAR_DST_OVERFLOW_INTR_CLR** | WO | FAR Destination Overflow Interrupt Clear<br>Writing 1 to clear the FAR destination overflow interrupt. |
| **15**<br>**FAR_DST_ON_FLASH_INTR_CLR** | WO | FAR Destination No Flash Interrupt Clear<br>Writing 1 to clear the FAR destination no flash interrupt. |
| **14**<br>**REGION2_D_CROS_INTR_CLR** | WO | Region 2 Destination Cross Interrupt Clear<br>Writing 1 to clear the region 2 destination cross interrupt. |
| **13**<br>**REGION2_S_CROS_INTR_CLR** | WO | Region 2 Source Cross Interrupt Clear<br>Writing 1 to clear the region 2 source cross interrupt. |
| **12**<br>**REGION2_SIZE_INTR_CLR** | WO | Region 2 Size Interrupt Clear<br>Writing 1 to clear the region 2 size interrupt. |
| **11**<br>**REGION2_DST_INTR_CLR** | WO | Region 2 Destination Interrupt Clear<br>Writing 1 to clear the region 2 destination interrupt. |
| **10**<br>**REGION2_SRC_INTR_CLR** | WO | Region 2 Source Interrupt Clear<br>Writing 1 to clear the region 2 source interrupt. |
| **9**<br>**REGION1_D_CROS_INTR_CLR** | WO | Region 1 Destination Cross Interrupt Clear<br>Writing 1 to clear the region 1 destination cross interrupt. |
| **8**<br>**REGION1_S_CROS_INTR_CLR** | WO | Region 1 Source Cross Interrupt Clear<br>Writing 1 to clear the region 1 source cross interrupt. |

| | | |
|---|---|---|
| **7**<br>**REGION1_SIZE_INTR_CLR** | WO | Region 1 Size Interrupt Clear<br>Writing 1 to clear the region 1 size interrupt. |
| **6**<br>**REGION1_DST_INTR_CLR** | WO | Region 1 Destination Interrupt Clear<br>Writing 1 to clear the region 1 destination interrupt. |
| **5**<br>**REGION1_SRC_INTR_CLR** | WO | Region 1 Source Interrupt Clear<br>Writing 1 to clear the region 1 source interrupt. |
| **4**<br>**REGION0_D_CROS_INTR_CLR** | WO | Region 1 Destination Cross Interrupt Clear<br>Writing 1 to clear the region 0 destination cross interrupt. |
| **3**<br>**REGION0_S_CROS_INTR_CLR** | WO | Region 0 Source Cross Interrupt Clear<br>Writing 1 to clear the region 0 source cross interrupt. |
| **2**<br>**REGION0_SIZE_INTR_CLR** | WO | Region 0 Size Interrupt Clear<br>Writing 1 to clear the region 0 size interrupt. |
| **1**<br>**REGION0_DST_INTR_CLR** | WO | Region 0 Destination Interrupt Clear<br>Writing 1 to clear the region 0 destination interrupt. |
| **0**<br>**REGION0_SRC_INTR_CLR** | WO | Region 0 Source Interrupt Clear<br>Writing 1 to clear the region 0 source interrupt. |

### 30.6.16 AHB_OVERLAY_INTR_EN (Interrupt Enable Register)

Offset : 8Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | INTR_ENABLE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_INTR_EN field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**INTR_ENABLE** | RW | Enable of Interrupt |

### 30.6.17 AHB_OVERLAY_FAR_SRC (FAR Source Register)

Offset : 90h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FAR_SRC_ADDR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FAR_SRC_ADDR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_FAR_SRC field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**FAR_SRC_ADDR** | RO | Source of FAR |

### 30.6.18 AHB_OVERLAY_FAR_DST (FAR Destination Register)

Offset : 94h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FAR_DST_ADDR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_FAR_DST field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-16**<br>**FAR_DST_ADDR** | RW | Destination of FAR<br>15:0 is fixed 16'b0 |

### 30.6.19 AHB_OVERLAY_FAR_SIZE (FAR Size Register)

Offset : 98h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | FAR_SIZE_VAL | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_FAR_SIZE field descriptions**

| Field | Type | Description |
|---|---|---|
| **8-0**<br>**FAR_SIZE_VAL** | RW | FAR Size<br>(FAR_SIZE_VAL+1)x64KB is the FAR size. |

### 30.6.20 AHB_OVERLAY_REGION0_MASK (Region 0 Mask Register)

Offset : A0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REGION0_MASK | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REGION0_MASK | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_REGION0_MASK field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-0**<br>**REGION0_MASK** | RO | Size Mask<br>The corresponding mask of Region 0 size. The mask is used to determine whether the input address is within the region range. The mask is calculated by the region size.<br>The corresponding relationship between size and mask is as follows:<br>1KB: 0xFFFF_FC00;<br>2KB: 0xFFFF_F800;<br>4KB: 0xFFFF_F000;<br>8KB: 0xFFFF_E000;<br>16KB: 0xFFFF_C000;<br>32KB: 0xFFFF_8000;<br>64KB: 0xFFFF_0000 |

### 30.6.21 AHB_OVERLAY_REGION1_MASK (Region 1 Mask Register)

Offset : A4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REGION1_MASK | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REGION1_MASK | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_REGION1_MASK field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-0**<br>**REGION1_MASK** | RO | Size Mask<br>The corresponding mask of Region 1 size. The mask is used to determine whether the input address is within the region range. The mask is calculated by the region size.<br>The corresponding relationship between size and mask is as follows:<br>1KB: 0xFFFF_FC00;<br>2KB: 0xFFFF_F800;<br>4KB: 0xFFFF_F000;<br>8KB: 0xFFFF_E000;<br>16KB: 0xFFFF_C000;<br>32KB: 0xFFFF_8000;<br>64KB: 0xFFFF_0000 |

### 30.6.22 AHB_OVERLAY_REGION2_MASK (Region 2 Mask Register)

Offset : A8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REGION2_MASK | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REGION2_MASK | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AHB_OVERLAY_REGION2_MASK field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-0**<br>**REGION2_MASK** | RO | Size Mask<br>The corresponding mask of Region 2 size. The mask is used to determine whether the input address is within the region range. The mask is calculated by the region size.<br>The corresponding relationship between size and mask is as follows:<br>1KB: 0xFFFF_FC00;<br>2KB: 0xFFFF_F800;<br>4KB: 0xFFFF_F000;<br>8KB: 0xFFFF_E000;<br>16KB: 0xFFFF_C000;<br>32KB: 0xFFFF_8000;<br>64KB: 0xFFFF_0000 |

Flagchip

# Chapter 31 System Clock Generator (SCG)

## 31.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • Added a new bit field "DONE[6]" to the register section *SCG_CRCCSR;*<br>• In the register section *SCG_CRCRES:*<br>- Changed the access attribute from "RO" to "RW";<br>- Updated the description of the "RESULT" bit field.<br>• In the register sections *SCG_SIRCTCFG* and *SCG_FIRCTCFG:*<br>- Updated the description of the "TRIMDIV" bit field;<br>- Added new bit fields "SAMPLE[3]" and "DELAY[2]" |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 31.2 Chip-specific Information

N/A

## 31.3 Features

Key features of the System Clock Generator (SCG) module are:

- 2 Phase-Locked Loops (PLL0/PLL1):
  - Voltage-controlled oscillator (VCO)
  - Selectable Internal or External reference clock is used as the PLL reference clock source
  - VCO frequency divider
  - Phase/Frequency detector
  - Integrated loop filter
  - PLL0 can be selected as the clock source for the MCU system clocks
  - 3 programmable post-dividers clock outputs, which can be used as clock sources for other on-chip peripherals
- 3 Internal Reference Clock (IRC) generators:
  - Fast IRC clock with 96 MHz (FIRC), Slow IRC clock with 12 MHz (SIRC), and Slow IRC clock with 32 kHz (SIRC32K).
  - FIRC 48 MHz (divided from FIRC) can be used as a source of the PLL reference clock
  - FIRC 96 MHz/ SIRC 12 MHz can be selected as the clock source for the MCU system clocks
  - 3 programmable post-divider clock outputs for FIRC and SIRC, which can be used as clock sources for other on-chip peripherals
  - FIRC and SIRC support automatic trim function, can select either FOSC or SOSC as the reference clock.
- 2 Crystal Oscillators:
  - FOSC clock with 40 MHz (typical) and SOSC clock with 32 kHz
  - FOSC can be used as the source of PLL reference clock
  - FOSC can be selected as the clock source for the MCU system clocks
  - 3 programmable post-divider clock outputs for FOSC, which can be used as clock sources for other on-chip peripherals
- Internal Clock Monitor:
  - Clock monitor with reset and interrupt request capability for PLL0, PLL1, FOSC, and SOSC
    - Lock/Unlock detector with reset and interrupt request capability for PLL0 and PLL1
  - Clock monitor with interrupt request capability for FIRC/SIRC automatic trim error
  - System slow clock monitor. If the clock is lost, the system will reset
- Function Safety:
  - Internal watchdog for entering and exiting low power mode. If timed out, the system will be reset
  - Support CRC logic for registers and generating a CRC error signal that can be processed by the FCSMU
  - If the external clock monitor (CMU4) generates an error, the SCG will cause the system to generate a reset

## 31.4 Overview

The System Clock Generator (SCG) module provides the system clocks for the MCU. The SCG contains two phase-locked loop (PLL0/1), two slow internal reference clocks (SIRC and SIRC32K), a fast internal reference clock (FIRC), a slow oscillator clock (SOSC), and a fast oscillator clock (FOSC). The PLL is sourced by either the FOSC reference clock or the FIRC 48 MHz reference clock. The SCG can select either PLL0, FOSC, or FIRC as the source for the MCU system clocks. The SCG also supports operation with crystal oscillators, allowing an external crystal or external clock source to produce the external reference clock.

***Note:*** If a PLL0 loss of lock occurs, the SCG will reduce the frequency of the output clock of the PLL0 to half optionally, so as to prevent program runaway caused by too high frequency from happening.

## 31.5 Functional Description

After resetting, the SCG boots with FIRC 96 MHz as the system clock source.

### 31.5.1 Clock Enable in Run Mode

| *Clock* | *Config Sequence* |
|---|---|
| *FIRC* | 1) Configure FIRCCSR[EN] to 1<br>2) Wait for FIRCCSR[VLD] to equal 1 |
| *FOSC* | 1) Configure FOSCCSR[EN] to 1<br>2) Wait for FOSCCSR[VLD] to equal 1 |
| *PLL0* | Make sure that the source clock is enabled and valid before enabling PLL0.<br>1) Enable source clock<br>2) Configure PLL0CFG register<br>3) Configure PLL0CSR[EN] to 1<br>4) Wait for PLL0CSR[VLD] to equal 1 |
| *PLL1* | Make sure that the source clock is enabled and valid before enabling PLL1.<br>1) Enable source clock.<br>2) Configure PLL1CFG register<br>3) Configure PLL1CSR[EN] to 1<br>4) Wait for PLL1CSR[VLD] to equal 1 |
| *SIRC* | SIRC is always on in run mode. |
| *SOSC* | 1) Configure SOSCCSR[EN] to 1<br>2) Wait for SOSCCSR[VLD] to equal 1 |
| *SIRC32K* | 1) Configure SIRC32KCSR[EN] to 1<br>2) Wait for SIRC32KCSR[VLD] to equal 1 |

### 31.5.2 Clock Status in Low Power Mode

If all clock sources are enabled, the status of all clock sources in low power mode is as follows:

| | *SIRC* | *FIRC* | *FOSC* | *PLL0* | *PLL1* | *SOSC* | *SIRC32K* |
|---|---|---|---|---|---|---|---|
| ***Run*** | ON | ON | ON | ON | ON | ON | ON |
| ***Stop*** | Configurable | Configurable | Configurable | Configurable | Configurable | ON | ON |
| ***Standby*** | Configurable | OFF | OFF | OFF | OFF | ON | ON |

The configurable clocks can be turned on or off via CSR[LPEN] and CSR[STEN] bits of the CSR register corresponding to the clock.

### 31.5.3 Function Clock Configuration

FIRC/FOSC/PLL0/PLL1/SIRC has 3 function clock divider outputs, which can be used as clock sources for other peripherals. The function clocks can be configured through the corresponding DIV register. The three divider outputs are named DIVH, DIVM, and DIVL. The clock output after DIVH should be not above 150 MHz, DIVM should be not above 150 MHz, and DIVL should be not above 75 MHz.

***Note:*** When configuring the corresponding function clock, set DIV*_EN to 0 first (* indicates H, M, or L).

For example, the clock configuration sequence for "FIRC_DIVH_CLK" is as follows:

1) Clear FIRCDIV[DIVH_EN], then wait for FIRCDIV[DIVH_ACK] to be cleared.
2) Configure FIRCDIV[DIVH].
3) Set FIRCDIV[DIVH_EN], then wait for FIRCDIV[DIVH_ACK] to be set. The configuration will be completed after 1 FIRC_DIVH_CLK cycle.

### 31.5.4 System Clock Switch Sequence

To switch the clock source from a slow clock to a fast clock (e.g., FOSC to PLL0):

1) Turn on the clock you want to switch (PLL0, FIRC, or FOSC).
2) Configure CCR register to select the clock source and divider.
3) Check CSR register, then wait for CSR[CCR_UPRD] = 1 and CSR[SCS] = CCR[SCS].
4) Wait for *CSR[SEL] = 1 (* = next system clock source).

To switch the clock source from a fast clock to a slow clock (e.g., PLL0 to FOSC):

1) Turn on the clock you want to switch (PLL0, FIRC, or FOSC).
2) Configure CCR register to select the clock source.
3) Check CSR register, then wait for CSR[CCR_UPRD] = 1 and CSR[SCS] = CCR[SCS].
4) Wait *CSR[SEL] = 1 (* = next system clock source).
5) Configure CCR register to select the divider.
6) Check CSR register, then wait for CSR[CCR_UPRD] = 1.

***Note:*** If a clock is used as the system clock, the CSR[EN] of the clock cannot be disabled.

### 31.5.5 FIRC/SIRC Automatic Trim

Automatic Trim function will be enabled by setting *CSR[TREN] and *CSR[TRUP]. Then the SCG can automatically trim FIRC/SIRC to 96MHz/12MHz. Reading *STAT register can get the current trim value. *CSR[TRLK] reads 1 is the completion flag of automatic trim. ("*" means FIRC or SIRC.)

***Note:*** Do not write *STAT register when automatic trim is enabled.

### 31.5.6 NVM Clock Configuration

The NVM clock source can be switched by programming CLKOUTCFG[NVMCLK_FIRC] or CLKOUTCFG[NVMCLK_SIRC].

Before switching the NVM clock:

1) Disable NVMCLK_FIRC and NVMCLK_SIRC first.
2) Then set one of them you need, and the other must be 0. Note: It is not allowed to set them at the same time.

***Note:*** Before entering the Standby mode (with the FIRC disabled), it is required to switch the NVM clock source to the SIRC.

### 31.5.7 CMU4 Reference Clock Configuration

The CMU4 reference clock source can be switched by programming CLKOUTCFG[CMU4CLK_FOSC] or CLKOUTCFG[CMU4CLK_SIRC].

Before switching the CMU4 reference clock:

1) Disable CMU4CLK_FOSC and CMU4CLK_SIRC first.
2) Set one of them you need, and the other must be 0. ***Note:*** It is not allowed to set them at the same time.

### 31.5.8 Internal Watchdog

The SCG has internal low-power entry and exit WDOGs, the monitor error will trigger a system reset directly.

- For the low-power entry sleep WDOG, the timeout can be configured to SCG_SLPWDG[MSB]*256 + 255 SIRC cycles.
- For the low-power exit wakeup WDOG, the timeout can be configured to SCG_WKPWDG[MSB]*16384 + 16383 SIRC cycles.

***Note:*** If FOSC is used, the timeout period of the wakeup WDOG should be greater than the time period taken by FOSC changing from enabled to valid.

### 31.5.9 Configuration CRC Check

For detailed configuration, refer to *Section Configuration CRC Check* in *Chapter FCSMU*.

The SCG register CRC checklist is as follows:

**Table 63. CRC checklist of SCG registers**

| **Type** | **Registers** | | | |
|---|---|---|---|---|
| **FOSC-related** | FOSCCSR | FOSCDIV | FOSCCFG | - |
| **SIRC-related** | SIRCCSR | SIRCDIV | SIRCTCFG | - |
| **FIRC-related** | FIRCCSR | FIRCDIV | FIRCTCFG | FIRCCFG |
| **SOSC-related** | SOSCCSR | SOSCCFG | - | - |
| **PLL1-related** | PLL1CSR | PLL1DIV | PLL1CFG | - |
| **PLL0-related** | PLL0CSR | PLL0DIV | PLL0CFG | - |
| **SIRC32K-related** | SIRC32KCSR | - | - | - |
| **Others** | CCR | CLKOUTCFG | SLPWDG | WKPWDG |

Flagchip

## 31.6 Register Definition

SCG base address : 4002_3000h

**Register Overview - SCG**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **10h** | SCG_CSR (Clock Status Register) | 32 | RO | 8300_0010h | page |
| **14h** | SCG_CCR (Clock Control Register) | 32 | RW | 0300_0010h | page |
| **20h** | SCG_CLKOUTCFG (CLKOUT Configuration Register) | 32 | RW | 5000_0000h | page |
| **24h** | SCG_SLPWDG (Low Power Sleep WDOG Register) | 32 | RW | 8000_0000h | page |
| **28h** | SCG_WKPWDG (Low Power Wakeup WDOG Register) | 32 | RW | 8000_0000h | page |
| **30h** | SCG_CRCCSR (CRC Control Status Register) | 32 | RW | 0000_0000h | page |
| **34h** | SCG_CRCRES (CRC Result Register) | 32 | RW | 0000_0000h | page |
| **90h** | SCG_PLL_LOLC (PLL Loss of Lock Control Register) | 32 | RW | 0000_0000h | page |
| **100h** | SCG_FOSCCSR (Fast OSC Control Status Register) | 32 | RW | 0000_0000h | page |
| **104h** | SCG_FOSCDIV (Fast OSC Divide Register) | 32 | RW | 0000_0000h | page |
| **108h** | SCG_FOSCCFG (Fast OSC Configuration Register) | 32 | RW | 0104_C000h | page |
| **200h** | SCG_SIRCCSR (Slow IRC Control Status Register) | 32 | RW | 0100_0000h | page |
| **204h** | SCG_SIRCDIV (Slow IRC Divide Register) | 32 | RW | 0000_0000h | page |
| **20Ch** | SCG_SIRCTCFG (Slow IRC Auto Trim Configuration Register) | 32 | RW | 0000_0000h | page |
| **218h** | SCG_SIRCSTAT (Slow IRC Software Trim Status Register) | 32 | RW | 0000_2020h | page |
| **300h** | SCG_FIRCCSR (Fast IRC Control Status Register) | 32 | RW | 0300_0001h | page |
| **304h** | SCG_FIRCDIV (Fast IRC Divide Register) | 32 | RW | 0000_0000h | page |
| **308h** | SCG_FIRCCFG (Fast IRC Configuration Register) | 32 | RW | 0000_0003h | page |
| **30Ch** | SCG_FIRCTCFG (Fast IRC Auto Trim Configuration Register) | 32 | RW | 0000_0000h | page |
| **318h** | SCG_FIRCSTAT (Fast IRC Software Trim Status Register) | 32 | RW | 0000_3F80h | page |
| **400h** | SCG_SOSCCSR (Slow OSC Control Status Register) | 32 | RW | 0000_0000h | page |
| **404h** | SCG_SOSCCFG (Slow OSC Configuration Register) | 32 | RW | 0040_0080h | page |
| **500h** | SCG_PLL1CSR (PLL1 Control Status Register) | 32 | RW | 0000_0000h | page |
| **504h** | SCG_PLL1DIV (PLL1 Divide Register) | 32 | RW | 0000_0000h | page |
| **508h** | SCG_PLL1CFG (PLL1 Configuration Register) | 32 | RW | 0077_0F10h | page |
| **600h** | SCG_PLL0CSR (PLL0 Control Status Register) | 32 | RW | 0000_0000h | page |

| | | | | | |
|---|---|---|---|---|---|
| **604h** | SCG_PLL0DIV (PLL0 Divide Register) | 32 | RW | 0000_0000h | page |
| **608h** | SCG_PLL0CFG (PLL0 Configuration Register) | 32 | RW | 0077_0F10h | page |
| **700h** | SCG_SIRC32KCSR (32K Slow IRC Control Status Register) | 32 | RW | 0000_0000h | page |

### 31.6.1 SCG_CSR (Clock Status Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CCR_UPRD | 0 | | | SCS | | | | 0 | | | | | | DIVCORE | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | DIVBUS | | 0 | | DIVSLOW | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**SCG_CSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**CCR_UPRD** | RO | CCR Register Update Ready<br>0b - CCR register is updating<br>1b - CCR register update finished |
| **27-24**<br>**SCS** | RO | System Clock Source<br>Gets the currently configured clock system clock source.<br>0001b - FOSC_CLK<br>0010b - SIRC_CLK<br>0011b - FIRC_CLK<br>0110b - PLL0_CLK<br>Others - Reserved |
| **17-16**<br>**DIVCORE** | RO | Core Clock Divide Ratio<br>00b~11b - Divide-by-1~4 |
| **5-4**<br>**DIVBUS** | RO | Bus Clock Divide Ratio<br>00b~11b - Divide-by-1~4 |
| **1-0**<br>**DIVSLOW** | RO | Slow Clock Divide Ratio<br>00b~11b - Divide-by-1~4 |

### 31.6.2 SCG_CCR (Clock Control Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SYSCLK_CME | 0 | | | SCS | | | | 0 | | | | | | DIVCORE | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | DIVBUS | | 0 | | DIVSLOW | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**SCG_CCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**SYSCLK_CME** | RW | Bus Clock Monitor<br>This field will generate a reset if the bus clock is lost. Clock monitor is disabled if the clock source is disabled or system enters standby mode.<br>0b - Disable bus clock monitor<br>1b - Enable bus clock monitor |
| **27-24**<br>**SCS** | RW | System Clock Select<br>Selects a system clock source. Do not select the reserved setting which will cause unknown failure. Switching to a different clock source requires that the clock source to be enabled first and be valid before system clocks are allowed to switch to that clock source.<br>Note: This field can only be written if clock source is allowed and valid.<br>0001b - FOSC_CLK<br>0010b - SIRC_CLK<br>0011b - FIRC_CLK<br>0110b - PLL0_CLK<br>Others - Reserved |
| **17-16**<br>**DIVCORE** | RW | Core Clock Divide Ratio<br>00b~11b - Divide-by-1~4 |
| **5-4**<br>**DIVBUS** | RW | Bus Clock Divide Ratio<br>00b~11b - Divide-by-1~4 |
| **1-0**<br>**DIVSLOW** | RW | Slow Clock Divide Ratio<br>00b~11b - Divide-by-1~4 |

### 31.6.3 SCG_CLKOUTCFG (CLKOUT Configuration Register)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NVMCLK_FIRC | NVMCLK_SIRC | CMU4CLK_FOSC | CMU4CLK_SIRC | CLKOUTSEL | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_CLKOUTCFG field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**NVMCLK_FIRC** | RW | Select FIRC12M as NVM initial clock source<br>Note: Before switching this clock, disable NVMCLK_FIRC and NVMCLK_SIRC first, then set one of them you need, and the other must be 0. It is not allowed to set both at the same time.<br>0b - Not select FIRC12M<br>1b - Select FIRC12M |
| **30**<br>**NVMCLK_SIRC** | RW | Select SIRC as NVM initial clock source<br>Note: Before switching this clock, disable NVMCLK_FIRC and NVMCLK_SIRC first, then set one of them you need, and the other must be 0. It is not allowed to set both at the same time.<br>0b - Not select SIRC<br>1b - Select SIRC |
| **29**<br>**CMU4CLK_FOSC** | RW | Select FOSC as CMU4 reference clock source<br>Note: Before switching this clock, disable CMU4CLK_FOSC and CMU4CLK_SIRC first, then set one of them you need, and the other must be 0. It is not allowed to set both at the same time.<br>0b - Not select FOSC<br>1b - Select FOSC |

| | | |
|---|---|---|
| **28**<br>**CMU4CLK_SIRC** | RW | Select SIRC as CMU4 reference clock source<br>Note: Before switching this clock, disable CMU4CLK_FOSC and CMU4CLK_SIRC first, then set one of them you need, and the other must be 0. It is not allowed to set both at the same time.<br>0b - Not select SIRC<br>1b - Select SIRC |
| **27-24**<br>**CLKOUTSEL** | RW | SCG Output Clock Select<br>Select the output clock source to SCG_CLKOUT pin.<br>0000b - OFF<br>0001b - FOSC_CLK<br>0010b - SIRC_CLK<br>0011b - FIRC_CLK<br>0100b - SOSC_CLK<br>0101b - PLL1_CLK<br>0110b - PLL0_CLK<br>0111b - SIRC32K_CLK<br>Others - Reserved |

### 31.6.4 SCG_SLPWDG (Low Power Sleep WDOG Register)

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EN | 0 | | | | | | | MSB | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_SLPWDG field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**EN** | RW | Sleep WDOG Enable<br>0b - Disable sleep WDOG<br>1b - Enable sleep WDOG |
| **23-16**<br>**MSB** | RW | Most Significant Bit<br>Timeout counter, MSB*256 + 255 SIRC cycles. |

### 31.6.5 SCG_WKPWDG (Low Power Wakeup WDOG Register)

Offset : 28h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EN | 0 | | | | | | | MSB | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_WKPWDG field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**EN** | RW | Wakeup WDOG Enable<br>0b - Disable wakeup WDOG<br>1b - Enable wakeup WDOG |
| **23-16**<br>**MSB** | RW | Most Significant Bit<br>Timeout counter, MSB*16384 + 16383 SIRC cycles. |

### 31.6.6 SCG_CRCCSR (CRC Control Status Register)

Offset : 30h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | DONE | ERR | BUSY | EOEN | CHKEN | TRGEN | 0 |
| W | | | | | | | | | | w1c | w1c | | | | | GEN |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_CRCCSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **6**<br>**DONE** | W1C | CRC Done Flag<br>0b - CRC not done.<br>1b - CRC done. |
| **5**<br>**ERR** | W1C | CRC Error Flag<br>0b - No CRC error.<br>1b - CRC error is detected. |
| **4**<br>**BUSY** | RO | CRC Busy Indicator<br>0b - CRC generator is idle.<br>1b - CRC generator is busy. |
| **3**<br>**EOEN** | RW | Error Output Enable<br>0b - CRC error output disable<br>1b - CRC error output enable |
| **2**<br>**CHKEN** | RW | CRC Checker Enable<br>0b - Disable CRC checker<br>1b - Enable CRC checker |
| **1**<br>**TRGEN** | RW | Trigger Enable<br>0b - Disable trigger.<br>1b - Enable trigger, then trigger input can start CRC generation. |
| **0**<br>**GEN** | WO | CRC Gnerate<br>0b - Do nothing.<br>1b - Start CRC generation. |

### 31.6.7 SCG_CRCRES (CRC Result Register)

Offset : 34h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RESULT | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RESULT | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_CRCRES field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**RESULT** | RW | CRC Result Value<br>This field stores the CRC result generated.<br>When CRCCSR[CHKEN] is set, reading this register gets the CRC result. It's a read-only register.<br>When CRCCSR[CHKEN] is cleared, writing this register can configure the golden CRC value. Reading this register can get the golden CRC value. |

### 31.6.8 SCG_PLL_LOLC (PLL Loss of Lock Control Register)

Offset : 90h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | PLL1_LOLC | PLL0_LOLC |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_PLL_LOLC field descriptions**

| Field | Type | Description |
|---|---|---|
| **1**<br>**PLL1_LOLC** | RW | PLL1 Behaviour after Loss of Lock<br>0b - Switch to half-frequency automatically<br>1b - Do not switch to half-frequency |
| **0**<br>**PLL0_LOLC** | RW | PLL0 Behaviour after Loss of Lock<br>0b - Switch to half-frequency automatically<br>1b - Do not switch to half-frequency |

### 31.6.9 SCG_FOSCCSR (Fast OSC Control Status Register)

Offset : 100h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ERR | SEL | VLD | LK | 0 | | | | | CMRE | CM |
| W | | | | | | w1c | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | STEN | EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_FOSCCSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **26**<br>**ERR** | W1C | Clock Error<br>This flag is reset on Chip POR only.<br>0b - Clock monitor is disabled or has not detected an error<br>1b - Clock monitor is enabled and has detected an error |
| **25**<br>**SEL** | RO | Clock Selected<br>0b - Clock is not selected as the system clock<br>1b - Clock is selected as the system clock |
| **24**<br>**VLD** | RO | Clock Valid<br>0b - Clock is disabled or invalid<br>1b - Clock is enabled and valid |
| **23**<br>**LK** | RW | Lock Register<br>This bit field can be written at any time.<br>0b - Register can be written<br>1b - Register cannot be written |
| **17**<br>**CMRE** | RW | Clock Monitor Reset Enable<br>This bit only works when CM is set.<br>0b - Generates an interrupt when an error detected<br>1b - Generates a reset when an error detected |
| **16**<br>**CM** | RW | Clock Monitor<br>Clock monitor cannot be enabled until VLD is set. Clock monitor is disabled if the clock source is disabled or system enters standby mode.<br>0b - Clock monitor is disabled<br>1b - Clock monitor is enabled |
| **1**<br>**STEN** | RW | Clock Stop Enable<br>0b - Clock is disabled in Stop mode<br>1b - Clock is enabled in Stop mode |
| **0**<br>**EN** | RW | Clock Enable<br>0b - Clock is disabled<br>1b - Clock is enabled |

### 31.6.10 SCG_FOSCDIV (Fast OSC Divide Register)

Offset : 104h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DIVL_ACK | DIVM_ACK | DIVH_ACK | 0 | DIVL_EN | DIVM_EN | DIVH_EN | 0 | | | | | DIVL | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | DIVM | | | 0 | | | | | DIVH | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_FOSCDIV field descriptions**

| Field | Type | Description |
|---|---|---|
| **30**<br>**DIVL_ACK** | RO | DIVL Function Clock Enable Ack<br>0b - DIVL function clock has been disabled<br>1b - DIVL function clock has been enabled |
| **29**<br>**DIVM_ACK** | RO | DIVM Function Clock Enable Ack<br>0b - DIVM function clock has been disabled<br>1b - DIVM function clock has been enabled |
| **28**<br>**DIVH_ACK** | RO | DIVH Function Clock Enable Ack<br>0b - DIVH function clock has been disabled<br>1b - DIVH function clock has been enabled |
| **26**<br>**DIVL_EN** | RW | DIVL Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |
| **25**<br>**DIVM_EN** | RW | DIVM Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |
| **24**<br>**DIVH_EN** | RW | DIVH Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |

| | | |
|---|---|---|
| **18-16**<br>**DIVL** | RW | Divide Ratio of Low-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVL_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |
| **10-8**<br>**DIVM** | RW | Divide Ratio of Middle-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVM_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |
| **2-0**<br>**DIVH** | RW | Divide Ratio of High-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVH_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |

### 31.6.11 SCG_FOSCCFG (Fast OSC Configuration Register)

Offset : 108h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BYPASS | 0 | | | | | | COMP_EN | EOCV | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | GM_SEL | | | | ALC_D | HYST_D | 0 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_FOSCCFG field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**BYPASS** | RW | Oscillator Bypass<br>Configures FOSC to bypass mode. Fields COMP_EN and BYPASS specify the oscillator configuration.<br>0b - Internal oscillator not bypassed<br>1b - Internal oscillator bypassed |
| **24**<br>**COMP_EN** | RW | Comparator Enable<br>Power-down signal for main comparator. COMP_EN must be 1 when using an external crystal and 0 when FOSC is in input bypass mode. Fields COMP_EN and BYPASS specify the oscillator configuration.<br>0b - Comparator disable<br>1b - Comparator enable |
| **23-16**<br>**EOCV** | RW | End of Count Value<br>Specifies the end of the count value. The oscillator stabilization counter uses EOCV for comparison when turning the clock on. This counting period ensures that the external oscillator clock signal is stable before the system selects the clock as a source. The counter runs on the crystal clock divided by 4 and counts up to EOCV × 256. After the counting is finished, the FOSCCSR[VLD] bit will be set.<br>EOCV can only be configured when the clock is disabled. |
| **15-12**<br>**GM_SEL** | RW | Crystal GM Select<br>Selects the transconductance applied by the FOSC amplifier. This setting depends on crystal specification. FOSC will not function when GM_SEL = 0000b (0 transconductance). |
| **11**<br>**ALC_D** | RW | Auto Loop Control Disable<br>0b - ALC enable<br>1b - ALC disable |
| **10**<br>**HYST_D** | RW | Hysteresis Disable<br>0b - Comparator enable hysteresis<br>1b - Comparator disable hysteresis |

### 31.6.12 SCG_SIRCCSR (Slow IRC Control Status Register)

Offset : 200h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ERR | 0 | VLD | LK | 0 | | | | | | CM |
| W | | | | | | w1c | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | TRLK | TRUP | TREN | 0 | | | | | LPEN | STEN | 0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_SIRCCSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **26**<br>**ERR** | W1C | Clock Error<br>This flag is reset on Chip POR only.<br>0b - Clock monitor is disabled or has not detected an error<br>1b - Clock monitor is enabled and detected an error |
| **24**<br>**VLD** | RO | Clock Valid<br>0b - Clock is disabled or invalid<br>1b - Clock is enabled and valid |
| **23**<br>**LK** | RW | Lock Register<br>This bit field can be written at any time.<br>0b - Register can be written<br>1b - Register cannot be written |
| **16**<br>**CM** | RW | Clock Monitor<br>Clock monitor cannot be enabled until VLD is set. Clock monitor is disabled if the clock source is disabled or system enters standby mode.<br>0b - Clock monitor is disabled<br>1b - Clock monitor is enabled |
| **10**<br>**TRLK** | RO | IRC Clock Coarse Lock Status Flag |
| **9**<br>**TRUP** | RW | IRC Clock Trim Update Enable (auto trim) |
| **8**<br>**TREN** | RW | IRC Software Trim Enable |

| | | |
|---|---|---|
| **2**<br>**LPEN** | RW | Clock Standby Enable<br>0b - Clock is disabled in Standby modes<br>1b - Clock is enabled in Standby modes (also need STEN set) |
| **1**<br>**STEN** | RW | Clock Stop Enable<br>0b - Clock is disabled in Stop mode<br>1b - Clock is enabled in Stop mode |

### 31.6.13 SCG_SIRCDIV (Slow IRC Divide Register)

Offset : 204h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DIVL_ACK | DIVM_ACK | DIVH_ACK | 0 | DIVL_EN | DIVM_EN | DIVH_EN | 0 | | | | | DIVL | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | DIVM | | | 0 | | | | | DIVH | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_SIRCDIV field descriptions**

| Field | Type | Description |
|---|---|---|
| **30**<br>**DIVL_ACK** | RO | DIVL Function Clock Enable Ack<br>0b - DIVL function clock has been disabled<br>1b - DIVL function clock has been enabled |
| **29**<br>**DIVM_ACK** | RO | DIVM Function Clock Enable Ack<br>0b - DIVM function clock has been disabled<br>1b - DIVM function clock has been enabled |
| **28**<br>**DIVH_ACK** | RO | DIVH Function Clock Enable Ack<br>0b - DIVH function clock has been disabled<br>1b - DIVH function clock has been enabled |
| **26**<br>**DIVL_EN** | RW | DIVL Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |

| | | |
|---|---|---|
| **25**<br>**DIVM_EN** | RW | DIVM Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |
| **24**<br>**DIVH_EN** | RW | DIVH Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |
| **18-16**<br>**DIVL** | RW | Divide Ratio of Low-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVL_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |
| **10-8**<br>**DIVM** | RW | Divide Ratio of Middle-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVM_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |
| **2-0**<br>**DIVH** | RW | Divide Ratio of High-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVH_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |

### 31.6.14 SCG_SIRCTCFG (Slow IRC Auto Trim Configuration Register)

This register can only be written when SIRCCSR[TREN] is cleared.

Offset : 20Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | TRIMDIV | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | SAMPLE | DELAY | TRIMSRC | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_SIRCTCFG field descriptions**

| Field | Type | Description |
|---|---|---|
| **26-16**<br>**TRIMDIV** | RW | IRC Clock Auto Trim Reference Clock Divider<br>If the FOSC clock is selected as the reference clock, it needs to be divided into a 250kHz clock. If the SOSC clock is selected as the reference clock, TRIMDIV should be 0.<br>Divider = TRIMDIV bits + 1 |
| **3**<br>**SAMPLE** | RW | Autotrim Sample Control<br>0b - 3 cycles (used for sample) when SOSC used as the trim source; 14 cycles when FOSC used as the trim source.<br>1b - 5 cycles when SOSC used as the trim source; 28 cycles when FOSC used as the trim source |
| **2**<br>**DELAY** | RW | Autotrim Delay Control<br>0b - 1 cycle (used for delay) when SOSC used as the trim source; 40 cycles for FOSC used as the trim source.<br>1b - 2 cycles when FOSC used as the trim source; 80 cycles when FOSC used as the trim source. |
| **1-0**<br>**TRIMSRC** | RW | IRC Clock Auto Trim Reference Clock Source Select<br>00b - Reserved<br>01b - Reserved<br>10b - Selects FOSC as the reference clock<br>11b - Selects SOSC as the reference clock |

### 31.6.15 SCG_SIRCSTAT (Slow IRC Software Trim Status Register)

This register can only be written when SIRCCSR[TREN] is set and SIRCCSR[TRUP] is cleared.

Offset : 218h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | TRIMFINE | | | | | | 0 | | TRIMCOAR | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

**SCG_SIRCSTAT field descriptions**

| Field | Type | Description |
|---|---|---|
| **13-8**<br>**TRIMFINE** | RW | Software Fine Trim Value |
| **5-0**<br>**TRIMCOAR** | RW | Software Coarse Trim Value |

### 31.6.16 SCG_FIRCCSR (Fast IRC Control Status Register)

Offset : 300h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ERR | SEL | VLD | LK | 0 | | | | | | CM |
| W | | | | | | w1c | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | TRLK | TRUP | TREN | 0 | | | | | | STEN | EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**SCG_FIRCCSR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **26**<br>**ERR** | W1C | Clock Error<br>This flag is reset on Chip POR only.<br>0b - Clock Monitor is disabled or has not detected an error<br>1b - Clock Monitor is enabled and has detected an error |
| **25**<br>**SEL** | RO | Clock Selected<br>0b - Clock is not selected as the system clock<br>1b - Clock is selected as the system clock |
| **24**<br>**VLD** | RO | Clock Valid<br>0b - Clock is disabled or invalid<br>1b - Clock is enabled and valid |
| **23**<br>**LK** | RW | Lock Register<br>This bit field can be written at any time.<br>0b - Register can be written<br>1b - Register cannot be written |
| **16**<br>**CM** | RW | Clock Monitor<br>Clock monitor cannot be enabled until VLD is set. Clock monitor is disabled if the clock source is disabled or system enters standby mode.<br>0b - Clock Monitor is disabled<br>1b - Clock Monitor is enabled |
| **10**<br>**TRLK** | RO | IRC Clock Coarse Lock Status Flag |
| **9**<br>**TRUP** | RW | IRC Clock Trim Update Enable (auto trim) |
| **8**<br>**TREN** | RW | IRC Software Trim Enable |
| **1**<br>**STEN** | RW | Clock Stop Enable<br>0b - Clock is disabled in Stop mode<br>1b - Clock is enabled in Stop mode |
| **0**<br>**EN** | RW | Clock Enable<br>0b - Clock is disabled<br>1b - Clock is enabled |

### 31.6.17 SCG_FIRCDIV (Fast IRC Divide Register)

Offset : 304h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DIVL_ACK | DIVM_ACK | DIVH_ACK | 0 | DIVL_EN | DIVM_EN | DIVH_EN | 0 | | | | | DIVL | | |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | DIVM | | | 0 | | | | | DIVH | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_FIRCDIV field descriptions**

| Field | Type | Description |
|---|---|---|
| **30**<br>**DIVL_ACK** | RO | DIVL Function Clock Enable Ack<br>0b - DIVL function clock has been disabled<br>1b - DIVL function clock has been enabled |
| **29**<br>**DIVM_ACK** | RO | DIVM Function Clock Enable Ack<br>0b - DIVM function clock has been disabled<br>1b - DIVM function clock has been enabled |
| **28**<br>**DIVH_ACK** | RO | DIVH Function Clock Enable Ack<br>0b - DIVH function clock has been disabled<br>1b - DIVH function clock has been enabled |
| **26**<br>**DIVL_EN** | RW | DIVL Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |
| **25**<br>**DIVM_EN** | RW | DIVM Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |
| **24**<br>**DIVH_EN** | RW | DIVH Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |

| | | |
|---|---|---|
| **18-16**<br>**DIVL** | RW | Divide Ratio of Low-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVL_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |
| **10-8**<br>**DIVM** | RW | Divide Ratio of Middle-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVM_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |
| **2-0**<br>**DIVH** | RW | Divide Ratio of High-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVH_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |

### 31.6.18 SCG_FIRCCFG (Fast IRC Configuration Register)

FIRCCFG register can only be configured when FIRC is disabled.

Offset : 308h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | CLKEN | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**SCG_FIRCCFG field descriptions**

| Field | Type | Description |
|---|---|---|
| **1-0**<br>**CLKEN** | RW | FIRC Internal Clock Enable<br>0b - FIRC 96MHz clock output enable.<br>1b - FIRC 48MHz clock output enable; can be used as the PLL reference clock. |

### 31.6.19 SCG_FIRCTCFG (Fast IRC Auto Trim Configuration Register)

This register can only be written when FIRCCSR[TREN] is cleared.

Offset : 30Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | TRIMDIV | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | SAMPLE | DELAY | TRIMSRC | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_FIRCTCFG field descriptions**

| Field | Type | Description |
|---|---|---|
| **26-16**<br>**TRIMDIV** | RW | IRC Clock Auto Trim Reference Clock Divider<br>If the FOSC clock is selected as the reference clock, it needs to be divided into a 250kHz clock. If the SOSC clock is selected as the reference clock, TRIMDIV should be 0.<br>Divider = TRIMDIV bits + 1 |

| | | |
|---|---|---|
| **3**<br>**SAMPLE** | RW | Autotrim Sample Control<br>0b - 3 cycles (used for sample) when SOSC used as the trim source; 14 cycles when FOSC used as the trim source.<br>1b - 5 cycles when SOSC used as the trim source; 28 cycles when FOSC used as the trim source |
| **2**<br>**DELAY** | RW | Autotrim Delay Control<br>0b - 1 cycle (used for delay) when SOSC used as the trim source; 40 cycles for FOSC used as the trim source.<br>1b - 2 cycles when FOSC used as the trim source; 80 cycles when FOSC used as the trim source. |
| **1-0**<br>**TRIMSRC** | RW | IRC Clock Auto Trim Reference Clock Source Select<br>00b - Reserved<br>01b - Reserved<br>10b - Selects FOSC as the reference clock<br>11b - Selects SOSC as the reference clock |

### 31.6.20 SCG_FIRCSTAT (Fast IRC Software Trim Status Register)

This register can only be written when FIRCCSR[TREN] is set and FIRCCSR[TRUP] is cleared.

Offset : 318h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | TRIMCOAR | | | | | | TRIMFINE | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_FIRCSTAT field descriptions**

| Field | Type | Description |
|---|---|---|
| **13-8**<br>**TRIMCOAR** | RW | Software Coarse Trim Value |
| **7-0**<br>**TRIMFINE** | RW | Software Fine Trim Value |

### 31.6.21 SCG_SOSCCSR (Slow OSC Control Status Register)

Offset : 400h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ERR | 0 | VLD | LK | 0 | | | | | CMRE | CM |
| W | | | | | | w1c | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | BYPASS | EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

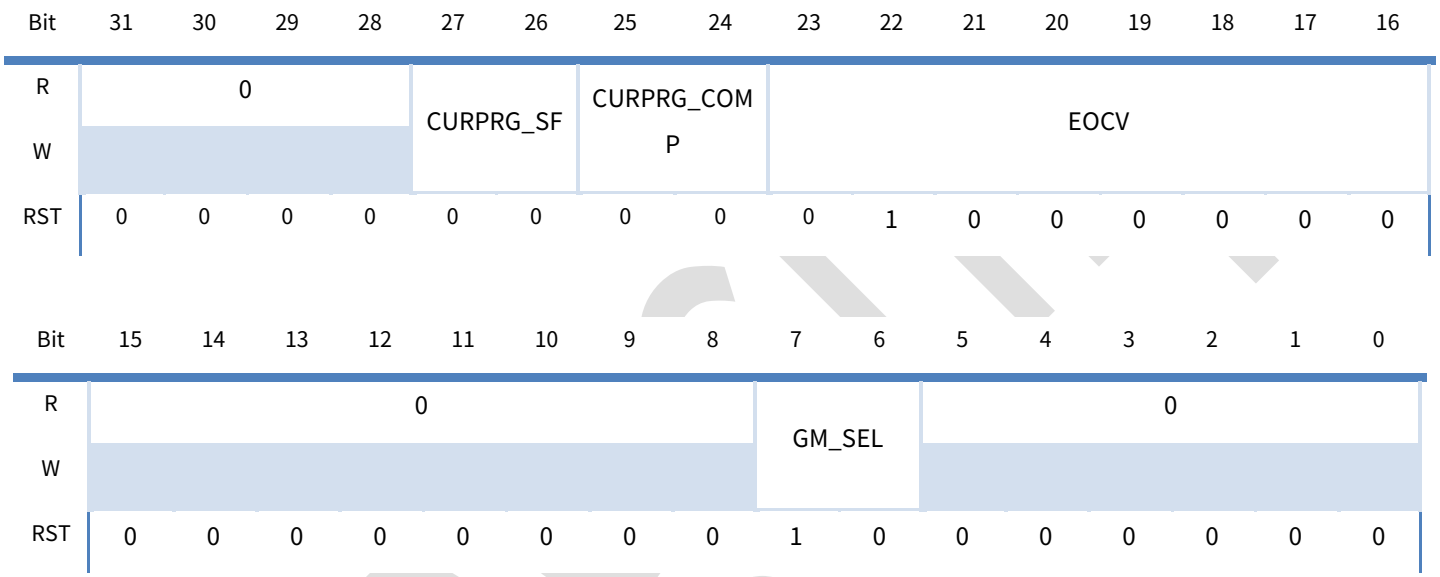
**SCG_SOSCCSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **26**<br>**ERR** | W1C | Clock Error<br>This flag is reset on Chip POR only.<br>0b - Clock Monitor is disabled or has not detected an error<br>1b - Clock Monitor is enabled and has detected an error |
| **24**<br>**VLD** | RO | Clock Valid<br>0b - Clock is disabled or invalid<br>1b - Clock is enabled and valid |
| **23**<br>**LK** | RW | Lock Register<br>This bit field can be written at any time.<br>0b - Register can be written<br>1b - Register cannot be written |
| **17**<br>**CMRE** | RW | Clock Monitor Reset Enable<br>This bit only works when CM is set.<br>0b - Generates an interrupt when an error detected<br>1b - Generates a reset when an error detected |
| **16**<br>**CM** | RW | Clock Monitor<br>Clock monitor cannot be enabled until VLD is set. Clock monitor is disabled if the clock source is disabled or system enters standby mode.<br>0b - Clock Monitor is disabled<br>1b - Clock Monitor is enabled |

| | | |
|---|---|---|
| **1**<br>**BYPASS** | RW | Oscillator Bypass<br>Configures SOSC to bypass mode. This bit can only be reset by POR.<br>0b - Internal oscillator not bypassed<br>1b - Internal oscillator bypassed |
| **0**<br>**EN** | RW | Clock Enable<br>This bit can only be reset by POR.<br>0b - Clock is disabled<br>1b - Clock is enabled |

### 31.6.22 SCG_SOSCCFG (Slow OSC Configuration Register)

Offset : 404h




**SCG_SOSCCFG field descriptions**

| Field | Type | Description |
|---|---|---|
| **27-26**<br>**CURPRG_SF** | RW | These bits specify programmability of level shifter current.<br>Recommended value: 11b |
| **25-24**<br>**CURPRG_COMP** | RW | These bits specify programmability of comparator current.<br>Recommended value: 11b |
| **23-16**<br>**EOCV** | RW | End of Count Value<br>Specify the end of the count value. This bit field can only be reset by POR. This value is used by the oscillator Stabilization counter for comparison whenever it is switched on. The oscillator counter runs on a crystal clock and counts values up to EOCV × 256. After the counting is finished, the SOSCCSR[VLD] bit will be set.<br>EOCV can only be configured when the clock is disabled. |

| | | |
|---|---|---|
| **7-6**<br>**GM_SEL** | RW | Crystal Overdrive Protection<br>This field setting decides the trans-conductance applied by the OSC amplifier, and it will depend on crystal specification.<br>Recommended value: 11b |

### 31.6.23 SCG_PLL1CSR (PLL1 Control Status Register)

Offset : 500h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | PLLLK | ERR | 0 | VLD | LK | 0 | | | | | CMRE | CM |
| W | | | | | | w1c | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | STEN | EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_PLL1CSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **27**<br>**PLLLK** | RO | PLL Clock Lock Status<br>0b - PLL clock has not been locked<br>1b - PLL clock has been locked |
| **26**<br>**ERR** | W1C | Clock Error<br>This flag is reset on Chip POR only.<br>0b - PLL Clock Monitor is disabled or has not detected an error<br>1b - PLL Clock Monitor is enabled and has detected an error |
| **24**<br>**VLD** | RO | PLL Valid<br>For PLL, valid is used to check if PLL is locked.<br>0b - PLL is not enabled or valid<br>1b - PLL is enabled and valid |
| **23**<br>**LK** | RW | Lock Register<br>This bit field can be written at any time.<br>0b - Control Status Register can be written<br>1b - Control Status Register cannot be written |

| | | |
|---|---|---|
| **17**<br>**CMRE** | RW | Clock Monitor Reset Enable<br>This bit only works when CM is set.<br>0b - Generates an interrupt when an error detected<br>1b - Generates a reset when an error detected |
| **16**<br>**CM** | RW | Clock Monitor<br>Clock monitor cannot be enabled until VLD is set. Clock monitor is disabled if the clock source is disabled or system enters standby mode.<br>0b - Clock Monitor is disabled<br>1b - Clock Monitor is enabled |
| **1**<br>**STEN** | RW | Clock Stop Enable<br>0b - Clock is disabled in Stop mode<br>1b - Clock is enabled in Stop mode |
| **0**<br>**EN** | RW | Clock Enable<br>0b - Clock is disabled<br>1b - Clock is enabled |

### 31.6.24 SCG_PLL1DIV (PLL1 Divide Register)

Offset : 504h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DIVL_ACK | DIVM_ACK | DIVH_ACK | 0 | DIVL_EN | DIVM_EN | DIVH_EN | 0 | | | | | DIVL | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | DIVM | | | 0 | | | | | DIVH | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_PLL1DIV field descriptions**

| Field | Type | Description |
|---|---|---|
| **30**<br>**DIVL_ACK** | RO | DIVL Function Clock Enable Ack<br>0b - DIVL function clock has been disabled<br>1b - DIVL function clock has been enabled |
| **29**<br>**DIVM_ACK** | RO | DIVM Function Clock Enable Ack<br>0b - DIVM function clock has been disabled<br>1b - DIVM function clock has been enabled |

| | | |
|---|---|---|
| **28**<br>**DIVH_ACK** | RO | DIVH Function Clock Enable Ack<br>0b - DIVH function clock has been disabled<br>1b - DIVH function clock has been enabled |
| **26**<br>**DIVL_EN** | RW | DIVL Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |
| **25**<br>**DIVM_EN** | RW | DIVM Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |
| **24**<br>**DIVH_EN** | RW | DIVH Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |
| **18-16**<br>**DIVL** | RW | Divide Ratio of Low-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVL_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |
| **10-8**<br>**DIVM** | RW | Divide Ratio of Middle-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVM_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |

| | | |
|---|---|---|
| **2-0**<br>**DIVH** | RW | Divide Ratio of High-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVH_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |

### 31.6.25 SCG_PLL1CFG (PLL1 Configuration Register)

PLLCFG can only be configured when PLL clock is disabled.

The PLL_CLK = Reference_Clock_Frequency / Pre-divider X Multiplier / Post-divider.

Reference_Clock is the clock source selected from the SOURCE bit of this register. Reference clock after PREDIV should be fall in the range of 2 MHz to 4 MHz.

The value of Pre-divider, Multiplier and Post-divider can refer to the description of the corresponding fields.

Offset : 508h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MULT | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | PREDIV | | | | | 0 | PSTDIV | | 0 | | | | SOURCE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**SCG_PLL1CFG field descriptions**

| Field | Type | Description |
|---|---|---|
| **24-16**<br>**MULT** | RW | PLL Multiplier<br>Multiplier = MULT bits + 1<br>Note: Multiplier needs to be greater than 96 (MULT needs to be greater than 95) |

| | | |
|---|---|---|
| **12-8**<br>**PREDIV** | RW | PLL Reference Clock Divider<br>Divide down the reference clock for PLL. Ensure that the resulting clock is within the required range of datasheet.<br>Pre-divider = PREDIV bits + 1<br>Note: Pre-divider range is 1 ~ 32 |
| **5-4**<br>**PSTDIV** | RW | PLL Post Divider<br>0b - Post-Divider divide 2<br>1b - Post-Divider divide 2<br>2b - Post-Divider divide 4<br>3b - Post-Divider divide 8 |
| **0**<br>**SOURCE** | RW | Clock Source<br>Configures the input clock source for PLL.<br>0b - Fast OSC (FOSC)<br>1b - Fast IRC (FIRC) with 2 frequency division |

### 31.6.26 SCG_PLL0CSR (PLL0 Control Status Register)

Offset : 600h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | PLLLK | ERR | SEL | VLD | LK | 0 | | | | | CMRE | CM |
| W | | | | | | w1c | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | STEN | EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_PLL0CSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **27**<br>**PLLLK** | RO | PLL Clock Lock Status<br>0b - PLL clock has not been locked<br>1b - PLL clock has been locked |
| **26**<br>**ERR** | W1C | Clock Error<br>This flag is reset on Chip POR only.<br>0b - PLL Clock Monitor is disabled or has not detected an error<br>1b - PLL Clock Monitor is enabled and has detected an error |

| | | |
|---|---|---|
| **25**<br>**SEL** | RO | Clock Selected<br>0b - Clock is not selected as the system clock<br>1b - Clock is selected as the system clock |
| **24**<br>**VLD** | RO | PLL Valid<br>For PLL, valid is used to check if PLL is locked.<br>0b - PLL is not enabled or valid<br>1b - PLL is enabled and valid |
| **23**<br>**LK** | RW | Lock Register<br>This bit field can be written at any time.<br>0b - Control Status Register can be written<br>1b - Control Status Register cannot be written |
| **17**<br>**CMRE** | RW | Clock Monitor Reset Enable<br>This bit only works when CM is set.<br>0b - Generates an interrupt when an error detected<br>1b - Generates a reset when an error detected |
| **16**<br>**CM** | RW | Clock Monitor<br>Clock monitor cannot be enabled until VLD is set. Clock monitor is disabled if the clock source is disabled or system enters standby mode.<br>0b - Clock Monitor is disabled<br>1b - Clock Monitor is enabled |
| **1**<br>**STEN** | RW | Clock Stop Enable<br>0b - Clock is disabled in Stop mode<br>1b - Clock is enabled in Stop mode |
| **0**<br>**EN** | RW | Clock Enable<br>Note: If set, this bit field cannot be disabled until VLD is set.<br>0b - Clock is disabled<br>1b - Clock is enabled |

### 31.6.27 SCG_PLL0DIV (PLL0 Divide Register)

Offset : 604h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DIVL_ACK | DIVM_ACK | DIVH_ACK | 0 | DIVL_EN | DIVM_EN | DIVH_EN | 0 | | | | | DIVL | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | DIVM | | | 0 | | | | | DIVH | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_PLL0DIV field descriptions**

| Field | Type | Description |
|---|---|---|
| **30**<br>**DIVL_ACK** | RO | DIVL Function Clock Enable Ack<br>0b - DIVL function clock has been disabled<br>1b - DIVL function clock has been enabled |
| **29**<br>**DIVM_ACK** | RO | DIVM Function Clock Enable Ack<br>0b - DIVM function clock has been disabled<br>1b - DIVM function clock has been enabled |
| **28**<br>**DIVH_ACK** | RO | DIVH Function Clock Enable Ack<br>0b - DIVH function clock has been disabled<br>1b - DIVH function clock has been enabled |
| **26**<br>**DIVL_EN** | RW | DIVL Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |
| **25**<br>**DIVM_EN** | RW | DIVM Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |
| **24**<br>**DIVH_EN** | RW | DIVH Function Clock Enable<br>0b - Disable function clock<br>1b - Enable function clock |
| **18-16**<br>**DIVL** | RW | Divide Ratio of Low-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVL_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |

| | | |
|---|---|---|
| **10-8**<br>**DIVM** | RW | Divide Ratio of Middle-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVM_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |
| **2-0**<br>**DIVH** | RW | Divide Ratio of High-frequency Peripheral Function Clock<br>Supply an asynchronous clock source.<br>Note: This field can only be written if DIVH_ACK is low<br>000b - Output disabled<br>001b - Divide-by-1<br>010b - Divide-by-2<br>011b - Divide-by-4<br>100b - Divide-by-8<br>101b - Divide-by-16<br>110b - Divide-by-32<br>111b - Divide-by-64 |

### 31.6.28 SCG_PLL0CFG (PLL0 Configuration Register)

PLLCFG can only be configured when PLL clock is disabled.

The PLL_CLK = Reference_Clock_Frequency / Pre-divider X Multiplier / Post-divider.

Reference_Clock is the clock source selected from the SOURCE bit of this register. Reference clock after PREDIV should be fall in the range of 2 MHz to 4 MHz.

For the values of Pre-divider, Multiplier and Post-divider, refer to the description of the corresponding fields.

Offset : 608h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | MULT | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | PREDIV | | | | | 0 | PSTDIV | | | 0 | | | SOURCE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**SCG_PLL0CFG field descriptions**

| Field | Type | Description |
|---|---|---|
| **24-16**<br>**MULT** | RW | PLL Multiplier<br>Multiplier = MULT bits + 1<br>Note: Multiplier needs to be greater than 96 (MULT needs to be greater than 95). |
| **12-8**<br>**PREDIV** | RW | PLL Reference Clock Divider<br>Divide down the reference clock for PLL. Ensure that the resulting clock is within the required range of datasheet.<br>Pre-divider = PREDIV bits + 1<br>Note: Pre-divider range is 1 ~ 32. |
| **5-4**<br>**PSTDIV** | RW | PLL Post Divider<br>0b - Post-Divider divide-by-2<br>1b - Post-Divider divide-by-2<br>2b - Post-Divider divide-by-4<br>3b - Post-Divider divide-by-8 |
| **0**<br>**SOURCE** | RW | Clock Source<br>Configures the input clock source for PLL.<br>0b - Fast OSC (FOSC)<br>1b - Fast IRC (FIRC) with 2 frequency division |

### 31.6.29 SCG_SIRC32KCSR (32K Slow IRC Control Status Register)

Offset : 700h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | VLD | LK | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SCG_SIRC32KCSR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **24**<br>**VLD** | RO | Clock Valid<br>0b - Clock is disabled or invalid<br>1b - Clock is enabled and valid |
| **23**<br>**LK** | RW | Lock Register<br>This bit can be written at any time, and can only be reset by POR.<br>0b - Register can be written<br>1b - Register cannot be written |
| **0**<br>**EN** | RW | Clock Enable<br>This bit can only be reset by POR.<br>0b - Clock is disabled<br>1b - Clock is enabled |

Flagchip

# Chapter 32 Peripheral Configuration Control (PCC)

## 32.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 32.2 Chip-specific Information

N/A

## 32.3 Features

The Peripheral Configuration Control (PCC) module supports the following configuration options for each peripheral:

- Clock gating control
- Selection of functional clock source
- Functional clock divider
- Software reset

***Note:*** The power domain that PCC registers reside in depends on the corresponding peripheral. The PCC of a peripheral belonging to PD0 does not need to be reconfigured after the chip is woken up from the STANDBY mode.

## 32.4 Overview

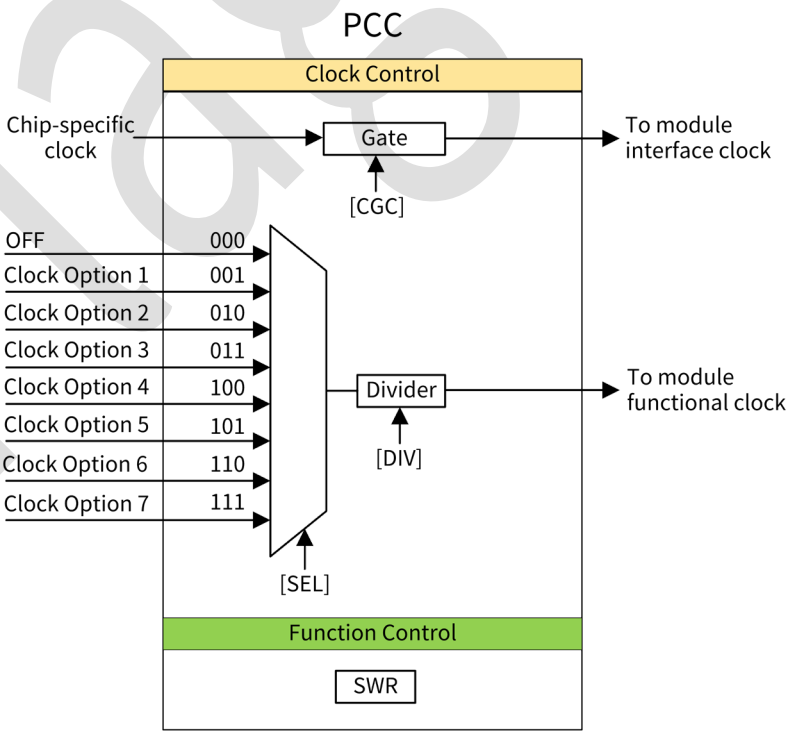
### 32.4.1 Peripheral Clock Control

The PCC module provides clock control and configuration for on-chip peripherals. Each PCC register contains a clock gating control (CGC) bit for a peripheral clock. The peripheral clock must be enabled in the PCC register (CGC = 1) corresponding to the peripheral before using the peripheral.

If a peripheral has a functional clock, the PCC register corresponding to the peripheral provides several clock source options through the Clock Source Select (SEL) field. In addition, the peripheral has a clock frequency divider by programming the Clock Divider Select (DIV) field. The peripheral clock must be disabled (CGC = 0) before configuring the functional clock.

Below is a block diagram of the PCC module:

**Figure 44. PCC block diagram**




Tips:
1. Not all module functional clocks have a divider
2. Interface clock = Internal bus interface clock for module registers and logic
3. Functional clock = Clock for module applications (not all modules have a functional clock)

### 32.4.2 Peripheral Reset Control

The PCC supports software resets of peripherals. When the Software Reset (SWR) field is set, the peripheral remains in the reset state. When the SWR field is cleared, a reset signal of the peripheral will be released.

***Note:*** Before resetting a peripheral by using the PCC, the NVIC Interrupt Set-Enable Register corresponding to the peripheral should be disabled.

## 32.5 Functional Description

Before changing the clock configuration of a peripheral, the user should ensure that the peripheral is disabled. For details, refer to *Chapter 9.2 Peripheral Clocks*.

## 32.6 Register Definition

PCC base address : 4002_4000h

**Register Overview - PCC**

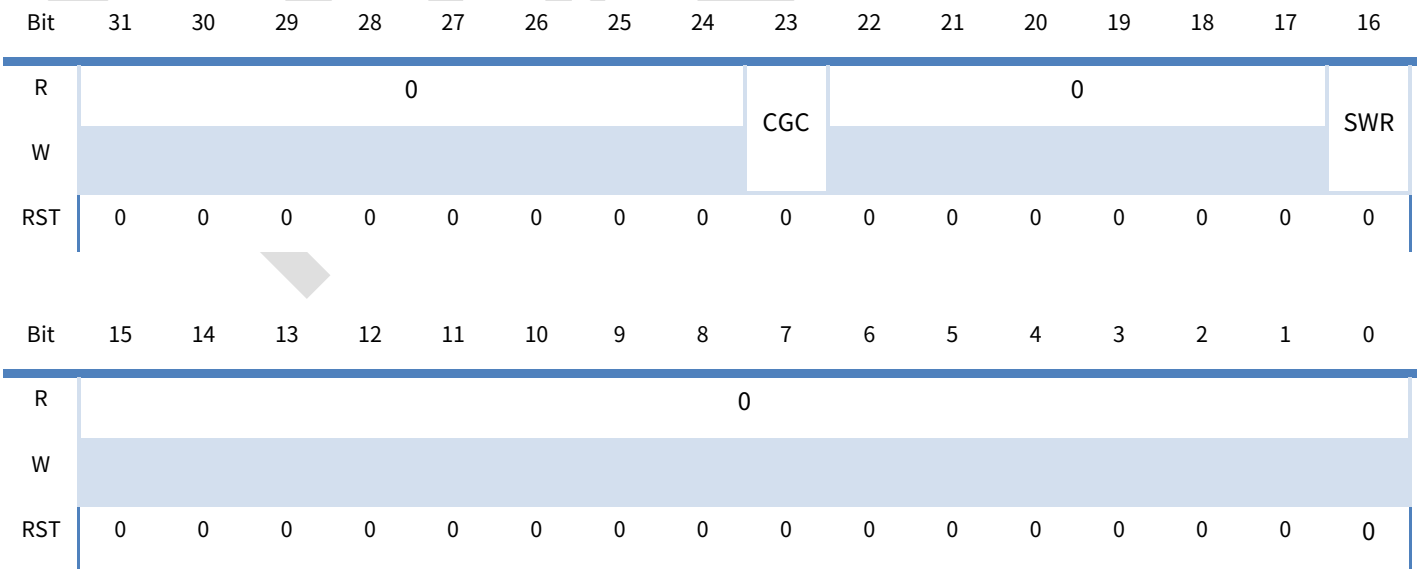
| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **20h** | PCC_DMA0 (DMA0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **28h** | PCC_DMAMUX0 (DMAMUX0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **4Ch** | PCC_ROMC (ROMC Clock Control Register) | 32 | RW | 0000_0000h | page |
| **60h** | PCC_ERM (ERM Clock Control Register) | 32 | RW | 0000_0000h | page |
| **64h** | PCC_EIM (EIM Clock Control Register) | 32 | RW | 0000_0000h | page |
| **68h** | PCC_INTM0 (INTM0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **6Ch** | PCC_ISM0 (ISM0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **88h** | PCC_WDOG0 (WDOG0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **98h** | PCC_TRGSEL0 (TRGSEL0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **9Ch** | PCC_TRGSEL1 (TRGSEL1 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **A0h** | PCC_TRGSEL2 (TRGSEL2 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **A4h** | PCC_TRGSEL3 (TRGSEL3 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **A8h** | PCC_CRC0 (CRC0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **ACh** | PCC_CORDIC0 (CORDIC0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **B0h** | PCC_TSTMP0 (TSTMP0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **B4h** | PCC_TSTMP1 (TSTMP1 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **B8h** | PCC_FCPIT0 (FCPIT0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **BCh** | PCC_AONTIMER0 (AONTIMER0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **C0h** | PCC_RTC (RTC Clock Control Register) | 32 | RW | 0000_0000h | page |
| **C4h** | PCC_CMU0 (CMU0 Clock Control Register) | 32 | RW | 0000_0000h | page |

| | | | | | |
|---|---|---|---|---|---|
| **C8h** | PCC_CMU1 (CMU1 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **CCh** | PCC_CMU2 (CMU2 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **D0h** | PCC_CMU3 (CMU3 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **D4h** | PCC_CMU4 (CMU4 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **DCh** | PCC_PTIMER0 (PTIMER0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **E0h** | PCC_PTIMER1 (PTIMER1 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **ECh** | PCC_ADC0 (ADC0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **F0h** | PCC_ADC1 (ADC1 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **FCh** | PCC_WKU (WKU Clock Control Register) | 32 | RW | 0000_0000h | page |
| **100h** | PCC_CMP0 (CMP0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **104h** | PCC_CMP1 (CMP1 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **10Ch** | PCC_TMU (TMU Clock Control Register) | 32 | RW | 0000_0000h | page |
| **150h** | PCC_SENT0 (SENT0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **160h** | PCC_MB (MB Clock Control Register) | 32 | RW | 0000_0000h | page |
| **170h** | PCC_FTU0 (FTU0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **174h** | PCC_FTU1 (FTU1 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **178h** | PCC_FTU2 (FTU2 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **17Ch** | PCC_FTU3 (FTU3 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **188h** | PCC_FCSPI0 (FCSPI0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **18Ch** | PCC_FCSPI1 (FCSPI1 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **190h** | PCC_FCSPI2 (FCSPI2 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **198h** | PCC_FCIIC0 (FCIIC0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **1A0h** | PCC_FCUART0 (FCUART0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **1A4h** | PCC_FCUART1 (FCUART1 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **1A8h** | PCC_FCUART2 (FCUART2 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **1ACh** | PCC_FCUART3 (FCUART3 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **1C0h** | PCC_LU0 (LU0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **1E0h** | PCC_FREQM (FREQM Clock Control Register) | 32 | RW | 0000_0000h | page |
| **1FCh** | PCC_STCU (STCU Clock Control Register) | 32 | RW | 0000_0000h | page |
| **200h** | PCC_FLEXCAN0 (FLEXCAN0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **210h** | PCC_FLEXCAN1 (FLEXCAN1 Clock Control Register) | 32 | RW | 0000_0000h | page |

| | | | | | |
|---|---|---|---|---|---|
| **34Ch** | PCC_WDOG1 (WDOG1 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **36Ch** | PCC_TRGSEL4 (TRGSEL4 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **370h** | PCC_TRGSEL5 (TRGSEL5 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **37Ch** | PCC_FCSPI3 (FCSPI3 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **380h** | PCC_FCSPI4 (FCSPI4 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **384h** | PCC_FCSPI5 (FCSPI5 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **3FCh** | PCC_FTU4 (FTU4 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **400h** | PCC_FTU5 (FTU5 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **404h** | PCC_FTU6 (FTU6 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **408h** | PCC_FTU7 (FTU7 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **41Ch** | PCC_FCIIC1 (FCIIC1 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **420h** | PCC_FCUART4 (FCUART4 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **424h** | PCC_FCUART5 (FCUART5 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **428h** | PCC_FCUART6 (FCUART6 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **42Ch** | PCC_FCUART7 (FCUART7 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **450h** | PCC_MSC0 (MSC0 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **480h** | PCC_FLEXCAN2 (FLEXCAN2 Clock Control Register) | 32 | RW | 0000_0000h | page |
| **490h** | PCC_FLEXCAN3 (FLEXCAN3 Clock Control Register) | 32 | RW | 0000_0000h | page |

### 32.6.1 PCC_DMA0 (DMA0 Clock Control Register)

Offset : 20h




**PCC_DMA0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.2 PCC_DMAMUX0 (DMAMUX0 Clock Control Register)

Offset : 28h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_DMAMUX0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.3 PCC_ROMC (ROMC Clock Control Register)

Offset : 4Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_ROMC field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.4 PCC_ERM (ERM Clock Control Register)

Offset : 60h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_ERM field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.5 PCC_EIM (EIM Clock Control Register)

Offset : 64h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_EIM field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.6 PCC_INTM0 (INTM0 Clock Control Register)

Offset : 68h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_INTM0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.7 PCC_ISM0 (ISM0 Clock Control Register)

Offset : 6Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_ISM0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.8 PCC_WDOG0 (WDOG0 Clock Control Register)

Offset : 88h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_WDOG0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.9 PCC_TRGSEL0 (TRGSEL0 Clock Control Register)

Offset : 98h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_TRGSEL0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.10 PCC_TRGSEL1 (TRGSEL1 Clock Control Register)

Offset : 9Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_TRGSEL1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.11 PCC_TRGSEL2 (TRGSEL2 Clock Control Register)

Offset : A0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_TRGSEL2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.12 PCC_TRGSEL3 (TRGSEL3 Clock Control Register)

Offset : A4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_TRGSEL3 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.13 PCC_CRC0 (CRC0 Clock Control Register)

Offset : A8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_CRC0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.14 PCC_CORDIC0 (CORDIC0 Clock Control Register)

Offset : ACh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_CORDIC0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.15 PCC_TSTMP0 (TSTMP0 Clock Control Register)

Offset : B0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_TSTMP0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.16 PCC_TSTMP1 (TSTMP1 Clock Control Register)

Offset : B4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_TSTMP1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.17 PCC_FCPIT0 (FCPIT0 Clock Control Register)

Offset : B8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCPIT0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.18 PCC_AONTIMER0 (AONTIMER0 Clock Control Register)

Offset : BCh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | DIV | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_AONTIMER0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |
| **2-0**<br>**DIV** | RW | Clock Divider Select<br>This field provides frequency division function for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Divide by 1.<br>001b - Divide by 2.<br>010b - Divide by 3.<br>011b - Divide by 4.<br>100b - Divide by 5.<br>101b - Divide by 6.<br>110b - Divide by 7.<br>111b - Divide by 8. |

### 32.6.19 PCC_RTC (RTC Clock Control Register)

Offset : C0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_RTC field descriptions**

| Field | Type | Description |
|---|---|---|
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.20 PCC_CMU0 (CMU0 Clock Control Register)

Offset : C4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_CMU0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.21 PCC_CMU1 (CMU1 Clock Control Register)

Offset : C8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_CMU1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.22 PCC_CMU2 (CMU2 Clock Control Register)

Offset : CCh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_CMU2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.23 PCC_CMU3 (CMU3 Clock Control Register)

Offset : D0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_CMU3 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.24 PCC_CMU4 (CMU4 Clock Control Register)

Offset : D4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_CMU4 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.25 PCC_PTIMER0 (PTIMER0 Clock Control Register)

Offset : DCh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_PTIMER0 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.26 PCC_PTIMER1 (PTIMER1 Clock Control Register)

Offset : E0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_PTIMER1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.27 PCC_ADC0 (ADC0 Clock Control Register)

Offset : ECh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | DIV | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_ADC0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |
| **2-0**<br>**DIV** | RW | Clock Divider Select<br>This field provides frequency division function for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Divide by 1.<br>001b - Divide by 2.<br>010b - Divide by 3.<br>011b - Divide by 4.<br>100b - Divide by 5.<br>101b - Divide by 6.<br>110b - Divide by 7.<br>111b - Divide by 8. |

### 32.6.28 PCC_ADC1 (ADC1 Clock Control Register)

Offset : F0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | DIV | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_ADC1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

| | | |
|---|---|---|
| **2-0**<br>**DIV** | RW | Clock Divider Select<br>This field provides frequency division function for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Divide by 1.<br>001b - Divide by 2.<br>010b - Divide by 3.<br>011b - Divide by 4.<br>100b - Divide by 5.<br>101b - Divide by 6.<br>110b - Divide by 7.<br>111b - Divide by 8. |

### 32.6.29 PCC_WKU (WKU Clock Control Register)

Offset : FCh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_WKU field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.30 PCC_CMP0 (CMP0 Clock Control Register)

Offset : 100h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_CMP0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.31 PCC_CMP1 (CMP1 Clock Control Register)

Offset : 104h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_CMP1 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.32 PCC_TMU (TMU Clock Control Register)

Offset : 10Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_TMU field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |

| | | |
|---|---|---|
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.33 PCC_SENT0 (SENT0 Clock Control Register)

Offset : 150h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | DIV | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_SENT0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

| | | |
|---|---|---|
| **2-0**<br>**DIV** | RW | Clock Divider Select<br>This field provides frequency division function for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Divide by 1.<br>001b - Divide by 2.<br>010b - Divide by 3.<br>011b - Divide by 4.<br>100b - Divide by 5.<br>101b - Divide by 6.<br>110b - Divide by 7.<br>111b - Divide by 8. |

### 32.6.34 PCC_MB (MB Clock Control Register)

Offset : 160h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_MB field descriptions**

| Field | Type | Description |
|---|---|---|
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.35 PCC_FTU0 (FTU0 Clock Control Register)

Offset : 170h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FTU0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - TCLK is selected.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.36 PCC_FTU1 (FTU1 Clock Control Register)

Offset : 174h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FTU1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - TCLK is selected.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.37 PCC_FTU2 (FTU2 Clock Control Register)

Offset : 178h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FTU2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - TCLK is selected.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.38 PCC_FTU3 (FTU3 Clock Control Register)

Offset : 17Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FTU3 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - TCLK is selected.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.39 PCC_FCSPI0 (FCSPI0 Clock Control Register)

Offset : 188h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCSPI0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.40 PCC_FCSPI1 (FCSPI1 Clock Control Register)

Offset : 18Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCSPI1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.41 PCC_FCSPI2 (FCSPI2 Clock Control Register)

Offset : 190h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCSPI2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.42 PCC_FCIIC0 (FCIIC0 Clock Control Register)

Offset : 198h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCIIC0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.43 PCC_FCUART0 (FCUART0 Clock Control Register)

Offset : 1A0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCUART0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.44 PCC_FCUART1 (FCUART1 Clock Control Register)

Offset : 1A4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCUART1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.45 PCC_FCUART2 (FCUART2 Clock Control Register)

Offset : 1A8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCUART2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.46 PCC_FCUART3 (FCUART3 Clock Control Register)

Offset : 1ACh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCUART3 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.47 PCC_LU0 (LU0 Clock Control Register)

Offset : 1C0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_LU0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.48 PCC_FREQM (FREQM Clock Control Register)

Offset : 1E0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FREQM field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.49 PCC_STCU (STCU Clock Control Register)

Offset : 1FCh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_STCU field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.50 PCC_FLEXCAN0 (FLEXCAN0 Clock Control Register)

Offset : 200h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | DIV | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FLEXCAN0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |
| **2-0**<br>**DIV** | RW | Clock Divider Select<br>This field provides frequency division function for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Divide by 1.<br>001b - Divide by 2.<br>010b - Divide by 3.<br>011b - Divide by 4.<br>100b - Divide by 5.<br>101b - Divide by 6.<br>110b - Divide by 7.<br>111b - Divide by 8. |

### 32.6.51 PCC_FLEXCAN1 (FLEXCAN1 Clock Control Register)

Offset : 210h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | DIV | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FLEXCAN1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |
| **2-0**<br>**DIV** | RW | Clock Divider Select<br>This field provides frequency division function for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Divide by 1.<br>001b - Divide by 2.<br>010b - Divide by 3.<br>011b - Divide by 4.<br>100b - Divide by 5.<br>101b - Divide by 6.<br>110b - Divide by 7.<br>111b - Divide by 8. |

### 32.6.52 PCC_WDOG1 (WDOG1 Clock Control Register)

Offset : 34Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_WDOG1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.53 PCC_TRGSEL4 (TRGSEL4 Clock Control Register)

Offset : 36Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_TRGSEL4 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.54 PCC_TRGSEL5 (TRGSEL5 Clock Control Register)

Offset : 370h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | 0 | | | | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_TRGSEL5 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.55 PCC_FCSPI3 (FCSPI3 Clock Control Register)

Offset : 37Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCSPI3 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.56 PCC_FCSPI4 (FCSPI4 Clock Control Register)

Offset : 380h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCSPI4 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.57 PCC_FCSPI5 (FCSPI5 Clock Control Register)

Offset : 384h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCSPI5 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.58 PCC_FTU4 (FTU4 Clock Control Register)

Offset : 3FCh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FTU4 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - TCLK is selected.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.59 PCC_FTU5 (FTU5 Clock Control Register)

Offset : 400h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FTU5 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - TCLK is selected.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.60 PCC_FTU6 (FTU6 Clock Control Register)

Offset : 404h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FTU6 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - TCLK is selected.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.61 PCC_FTU7 (FTU7 Clock Control Register)

Offset : 408h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FTU7 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - TCLK is selected.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.62 PCC_FCIIC1 (FCIIC1 Clock Control Register)

Offset : 41Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCIIC1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.63 PCC_FCUART4 (FCUART4 Clock Control Register)

Offset : 420h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCUART4 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.64 PCC_FCUART5 (FCUART5 Clock Control Register)

Offset : 424h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCUART5 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.65 PCC_FCUART6 (FCUART6 Clock Control Register)

Offset : 428h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCUART6 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.66 PCC_FCUART7 (FCUART7 Clock Control Register)

Offset : 42Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FCUART7 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral.<br>This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |

### 32.6.67 PCC_MSC0 (MSC0 Clock Control Register)

Offset : 450h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | DIV | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_MSC0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |

| | | |
|---|---|---|
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |
| **2-0**<br>**DIV** | RW | Clock Divider Select<br>This field provides frequency division function for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Divide by 1.<br>001b - Divide by 2.<br>010b - Divide by 3.<br>011b - Divide by 4.<br>100b - Divide by 5.<br>101b - Divide by 6.<br>110b - Divide by 7.<br>111b - Divide by 8. |

### 32.6.68 PCC_FLEXCAN2 (FLEXCAN2 Clock Control Register)

Offset : 480h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | DIV | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FLEXCAN2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |
| **2-0**<br>**DIV** | RW | Clock Divider Select<br>This field provides frequency division function for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Divide by 1.<br>001b - Divide by 2.<br>010b - Divide by 3.<br>011b - Divide by 4.<br>100b - Divide by 5.<br>101b - Divide by 6.<br>110b - Divide by 7.<br>111b - Divide by 8. |

### 32.6.69 PCC_FLEXCAN3 (FLEXCAN3 Clock Control Register)

Offset : 490h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CGC | SEL | | | 0 | | | SWR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | DIV | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PCC_FLEXCAN3 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**CGC** | RW | Clock Gate Control<br>This bit controls the clock for the peripheral.<br>0b - Clock disabled<br>1b - Clock enabled. Clock selection and divider fields are locked. |
| **22-20**<br>**SEL** | RW | Clock Source Select<br>This field selects different source for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Clock is off.<br>001b - Clock option 1<br>010b - Clock option 2<br>011b - Clock option 3<br>100b - Clock option 4<br>101b - Clock option 5<br>110b - Clock option 6<br>111b - Clock option 7 |
| **16**<br>**SWR** | RW | Software Reset<br>0b - Reset released.<br>1b - Reset held. |
| **2-0**<br>**DIV** | RW | Clock Divider Select<br>This field provides frequency division function for the functional clock of peripheral. This field can be written only when the clock is disabled (CGC = 0).<br>000b - Divide by 1.<br>001b - Divide by 2.<br>010b - Divide by 3.<br>011b - Divide by 4.<br>100b - Divide by 5.<br>101b - Divide by 6.<br>110b - Divide by 7.<br>111b - Divide by 8. |

# Chapter 33 Watchdog (WDOG)

## 33.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

This chapter describes the Watchdog (WDOG) Module. The WDOG contains the following sections:

- Chip-specific Information
- Introduction
- Features
- Functional Description
- Application Information
- Register Definition

## 33.2 Chip-specific Information

The details of the WDOG clock source selected by WDOG_CS[CLK_SEL] are as follows:

- WDOG_CS[CLK_SEL] = 00b: Bus clock
- WDOG_CS[CLK_SEL] = 01b: Always-on clock (AON_CLK)
- WDOG_CS[CLK_SEL] = 10b: Fast Oscillator clock (FOSC)
- WDOG_CS[CLK_SEL] = 11b: Internal Slow IRC clock (SIRC)

### 33.2.1 Clock Configuration

The clock source of WDOG is configurable and has the following options:

- Bus clock
- Always-on clock (AON_CLK)
- Fast Oscillator clock (FOSC)
- Internal Slow IRC clock (SIRC)

WDOG_COUNTER reset read value can vary depending on the timer stamp, since it is a default running counter.

### 33.2.2 Watchdog Timeout and Reaction

The WDOG counter will start counting on the selected clock source after initialization, and the counter timeout time depends on the clock source. After initial power-on, the clock source of the watchdog is SIRC 128K_CLK, and the watchdog will timeout after 1024 clock cycles. It takes about 8ms to generate the watchdog timeout interrupt which will force an MCU reset. To avoid the reset, ensure that the watchdog is configured or refreshed before the 1024 cycles.

The WDOG is configured to generate a system reset as the timeout reaction by default. Alternatively, it can be configured to generate an interrupt instead of a reset upon the counter timeout by configuring WDOG and Reset Generation Module (RGM). When configuring the timeout reaction to be an interrupt, the watchdog will delay the reset generation.

The interrupt reaction feature is controlled by WDOG.CS register in the WDOG and controlled by RGM.SRIE in the RGM. For the WDOG, the delay to generate the reset can be configured to 128, 256, 512, or 1024 bus clock cycles. To enable RGM IRQ, both SRIE[WDOG] and SRIE[GLOBAL_RIE] need to be set.

## 33.3 Overview

The WDOG module is an independent timer that provides a reliable and secure way to detect and recover from software or hardware failures for the system. Its main purpose is to ensure that the software executes as intended and the CPU is not stuck in an infinite loop or executing unexpected code. The WDOG module needs to be serviced (refreshed) within a certain period; otherwise, it will timeout and reset the MCU.

## 33.4 Features

The WDOG module includes the following features:

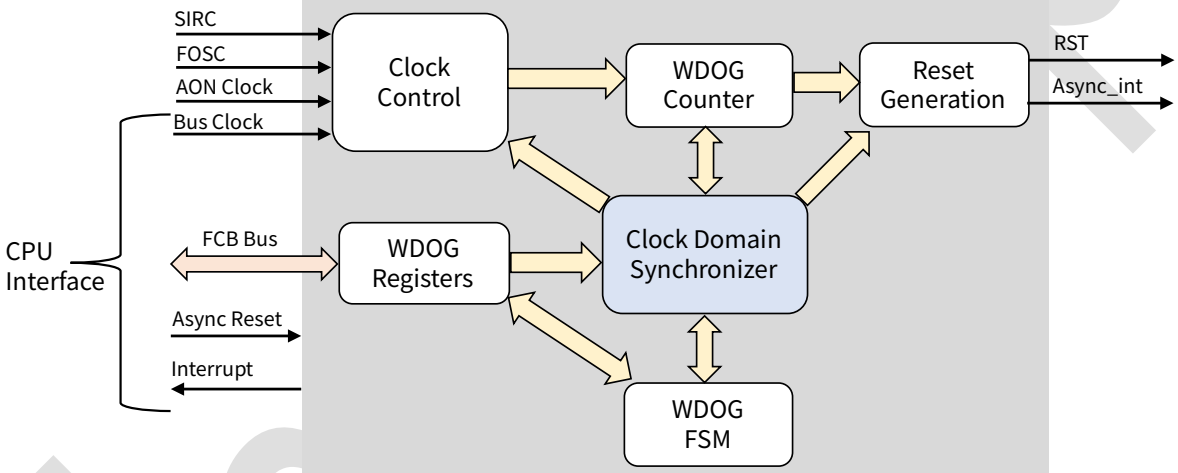
- Configurable input clock source and independent from the bus clock
- Programmable 16-bit timeout value composed of high 8 bits value and low 8 bits value in TIMEOUT register

- Implement an optional fixed 256 clock prescaler to get a larger timeout value
- Unlock sequence of writing 0x0818_1982 to allow updates of writeable once registers
- Software updates must be configured within 128 bus clocks after unlocking
- Refresh sequence of writing 0x20CF_FC20
- Programmable 16-bit window value composed of high 8 bits and low 8 bits in WINDOW register
- Refresh operation earlier than the window value will trigger a reset
- Optional timeout interrupt to allow post-processing diagnostics after a counter timeout occurs
- Delay of 128, 256, 512, or 1024 bus clocks to generate a reset after the interrupt vector fetched
- Registers related to watchdog configuration are write-once-after-reset to ensure that the configuration cannot be modified by mistake

## 33.5 Block Diagram

The following is the block diagram of the WDOG module.

**Figure 45. WDOG block diagram**




## 33.6 Functional Description

The WDOG module provides a reliable and secure mechanism to detect software or hardware failure and to ensure the system can be reset to a known state of operation from those situations. The watchdog counter runs continuously and independently on a selectable clock source and expects to be refreshed periodically; otherwise, it will generate a reset event.

### 33.6.1 Configuration of Watchdog

The control bits, timeout value, and window value in watchdog registers are writeable-once after reset so they cannot be changed until the next reset occurs. To guarantee this feature, the user should configure the window and timeout value first, then configure the other control bits. Ensure the CS[UPDATE] is set to 0 during the configuration. This is a secure and robust mechanism to configure the watchdog and prevent the configuration from being tampered by the runaway condition.

After all registers except COUNTER are written, the new configuration will take effect upon reset. Otherwise, the WDOG uses the reset values by default. If window mode is disabled (CS[WIN] is 0), the new configuration will take effect regardless of the WINDOW register value.

#### *33.6.1.1 Reconfiguration of Watchdog*

When the user needs to reconfigure or disable the watchdog without forcing a reset, the following methods are supported:

- Initialize the CS[UPDATE] to 1 in the first configuration after reset, and then execute the unlock sequence to reconfigure the watchdog at any time.

***Note:*** Any other reconfiguration after the first reconfiguration can only be operated when CS[RECFG_STAT] is 1.

- If CS[UPDATE] remains 0, reset is the only way to reconfigure the watchdog.

Execution of the unlock sequence: Writing 0x0818_1982 to the COUNTER register at any time after the watchdog has been configured will unlock it.

After completing the unlock sequence, the reconfiguration must be taken within 128 bus clocks; otherwise, the watchdog closes the unlock window.

Because the unlock window is 128 bus clock cycles, some delays must be inserted before executing STOP or WAIT instructions if the watchdog is just reconfigured. This ensures that the watchdog's new configuration takes effect before the MCU enters low power mode. Otherwise, the MCU may not be waken up from low power mode.

### 33.6.2 Watchdog Refresh Mechanism

The watchdog will timeout and reset the MCU if the watchdog counter is not refreshed. The watchdog refresh mechanism is secure and robust to prevent the counter from being tampered by the runaway code. To refresh the watchdog counter, a refresh write sequence needs to be executed by software before the timeout occurs. In addition, if window mode is enabled, the refresh sequence can only be started after the counter reached the minimum value set in the WINDOW register.

#### *33.6.2.1 Refresh Operation*

The refresh write sequence is a 32-bit write (0x20CF_FC20) to the COUNTER register. The sequence must be operated before the WDOG timeout; otherwise, the watchdog resets the MCU.

The global interrupts need to be disabled before executing the refresh sequence. Otherwise, an interrupt could invalidate the refresh sequence if the interrupt is generated before the refresh operation finish. After the sequence finishes, restore the global interrupt control state.

#### *33.6.2.2 Window Mode*

When the Window mode is enabled, the watchdog must be refreshed after the counter has reached a minimum expected time value; otherwise, the watchdog resets the MCU, any refresh attempts before the counter has reached a minimum expected time value will force a reset. Depending on the requirements of the application, the WDOG can be programmed to force a reset when refresh attempts are earlier than expected. The minimum expected time value is specified in the WINDOW register. Setting CS[WIN] enables window mode.

### 33.6.3 Interrupt Reaction Mode

The watchdog will generate an interrupt request immediately after a reset-triggering event occurs when interrupts are enabled (CS[INT] = 1). The reset will finally be generated by the watchdog after a delay of 128, 256, 512, or 1024 bus clock cycles. Different delay values are configured by CS[DLY_CNT_MSB], and the time window of the delay cycles is used for the interrupt service routine (ISR). The typical reset-triggering event is a counter timeout or invalid refresh attempt. When interrupts are disabled (CS[INT] = 0), the watchdog generates the reset upon the counter timeout.

### 33.6.4 Backup Reset

If the main WDOG logic loses its clock (the bus clock) and can no longer monitor the counter, the backup reset function will take charge of the generation for the reset. It is a safeguard measure and can work independently on the clock source except the bus clock. If the watchdog counter overflows twice continuously without generating any resets, the backup reset function will work and generate a reset.

To set up the backup reset function, the user needs to enable interrupt and configure the non-bus clock source for the watchdog. If an interrupt is enabled, and once the bus clock is cut off before exiting the interrupt service routine, the normal watchdog reset will be blocked. In this case, the second overflow will cause a backup reset directly.

### 33.6.5 Self-check Mechanism of Watchdog

To ensure the operation of the safety-critical applications, the user needs to check the function of the watchdog by checking every bit of the 16-bit counter.

Splitting the 16-bit counter into two byte-wide counters and letting them run independently will minimize the time of check. After counters timeout, compare the counter value with the corresponding timeout value. To check the overflow connection from the low byte to the high byte, feed the input clock via the 8$^{th}$ bit of the low byte when checking the high byte of the counter. To further speed up the checking, use a faster clock (such as the bus clock) for the counter reference.

On a power-on reset, the POR bit in the system reset register is set, indicating the user should perform the WDOG fast check.

#### *33.6.5.1 Checking Each Byte of Counter*

The steps to check the watchdog counter are as follows:

- Configure the watchdog timeout value in the TIMEOUT register.
- Configure CS[TST] = 10b to check the low byte of the counter; configure CS[TST] = 11b to check the high byte of the counter.
- Wait for the watchdog to time out and force a reset.
- Confirm that the WDOG flag in the system reset register is set, showing that the watchdog has generated the reset. (The POR flag remains clear)
- Confirm that CS[TST] shows the check for 10b or 11b was performed.

To check the low byte or high byte of the counter, compare the counter value with the timeout value for the selected bytes.

#### *33.6.5.2 User Mode*

After the counter self-check, the user can configure CS[TST] to 01b to indicate the watchdog is ready for application user mode. In this situation, the reset will be treated as a real watchdog reset caused by runaway or faulty application code by software.

During the counter checking with the default clock source, software can periodically read the CNT register to ensure the counter is increasing.

### 33.6.6 Clock Source

The watchdog supports several clock-source options which can be selected by configuring CS[CLK_SEL]. The details of available clock options and default configuration are described in the chip-specific section.

To meet more robust safety requirements, the clock sources independent of the bus clock can be selected for the applications. This feature will avoid the situation where the watchdog is using the bus clock while the clock is somehow halted.

After the functional reset of the watchdog is de-asserted, the default clock source for the watchdog should be enabled to ensure the proper function of the watchdog.

A fixed prescaler is implemented in the watchdog to achieve a larger timeout range for all clock sources. This feature is optional by configuring CS[PRESCALER]. If this bit is set, the clock source will be prescaled by 256 before clocking the watchdog counter.

In different configurations, the watchdog will have different timeout periods:

- If the prescaler is passed through, the available timeout range is (1 - 65535) × Reference Clock Period
- If the prescaler is enabled, the available timeout range is (256 - 16776960) × Reference Clock Period

When the programmer switches clock sources during reconfiguration, the watchdog counter keeps at zero for 2.5 cycles of the previous clock source and 2.5 cycles of the new clock source after the ending of the configuration time window (after 128 bus clocks). This delay ensures that the counter can be restarted with the new configuration smoothly.

## 33.7 Application Information

The watchdog is enabled by default after reset. It is suggested to disable or reconfigure the watchdog before the first watchdog timeout and at the very beginning of the software code, such as the beginning of the startup or the beginning of the main function.

To disable or reconfigure the watchdog without forcing a reset, WDOG_CS[UPDATE] bit must be set in the watchdog initial configuration and write the unlock sequence to the COUNTER register. After then, the user can disable or reconfigure the watchdog at any time within the timeout limit; to disable the watchdog at any time, operate the unlock sequence first, then unset the WDOG_CS[ENABLE] bit; to disable the watchdog after reset, just unset the WDOG_CS[ENABLE] bit, the unlock sequence is unnecessary because all watchdog registers are unlocked by reset.

To refresh the watchdog and reset the watchdog counter to zero, a refresh sequence is required.

## 33.8 Register Definition

WDOG0 base address : 4002_2000h

WDOG1 base address : 4043_3000h

**Register Overview - WDOG**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | WDOG_CS (Watchdog Control and Status Register) | 32 | RW | 0002_0980h | page |
| **4h** | WDOG_COUNTER (Watchdog Counter Register) | 32 | RW | 0000_0000h | page |
| **8h** | WDOG_TIMEOUT (Watchdog Timeout Value Register) | 32 | RW | 0000_0400h | page |
| **Ch** | WDOG_WINDOW (Watchdog Window Register) | 32 | RW | 0000_0000h | page |

### 33.8.1 WDOG_CS (Watchdog Control and Status Register)

This section describes the function of the Watchdog Control and Status Register. TST can only be cleared by POR, any other reset does not affect the value of this field. This register is writeable once after reset.

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | DLY_CNT_MSB | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | WIN | FLAG | 0 | PRESCALER | ULK_STAT | RECFG_STAT | CLK_SEL | ENABLE | INT | UPDATE | TST | | DBG | WAIT | STOP |
| W | | w1c | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**WDOG_CS field descriptions**

| Field | Type | Description |
|---|---|---|
| **17-16**<br>**DLY_CNT_MSB** | RW | Reset Delay Counter MSB<br>This field is used to add the value of the WDOG reset delay when enabling the WDOG interrupt. The default WDOG_RST_DLY is 512 bus clock cycles. This field is the left-shift value of the WDOG_RST_DLY.<br>This field should be configured by 32-bit access or configured before CS[15:8].<br>00b - Delay 128 bus clock cycles<br>01b - Delay 256 bus clock cycles<br>10b - Delay 512 bus clock cycles<br>11b - Delay 1024 bus clock cycles |
| **15**<br>**WIN** | RW | Window Mode<br>Enable window mode. This bit is writeable once after reset.<br>0b - Window mode disabled.<br>1b - Window mode enabled. |
| **14**<br>**FLAG** | W1C | Interrupt Flag<br>This bit indicates whether an interrupt happened. Write 1 to clear it.<br>0b - No interrupt occurred.<br>1b - An interrupt occurred. |
| **12**<br>**PRESCALER** | RW | Watchdog Prescaler<br>Enable a fixed 256 pre-scaling reference clock for watchdog counter. This bit is writeable once after reset.<br>0b - 256 prescaler disabled.<br>1b - 256 prescaler enabled. |
| **11**<br>**ULK_STAT** | RO | Unlock Status<br>This bit indicates whether WDOG is unlocked or not. This bit is read-only.<br>0b - WDOG is locked.<br>1b - WDOG is unlocked. |

| | | |
|---|---|---|
| **10**<br>**RECFG_STAT** | RO | Reconfiguration Status<br>This bit indicates the status of the WDOG reconfiguration. This bit is set when WDOG is successfully reconfigured, and will be cleared when unlock command is operated. This bit is read-only.<br>0b - Reconfigure action not implemented or not successful.<br>1b - Reconfiguration is successful. |
| **9-8**<br>**CLK_SEL** | RW | Watchdog Clock<br>These bits configure the clock source that will feed the watchdog counter. Detailed information is included in the chip-specific section. These bits are writeable once after reset.<br>00b - Select bus clk as watchdog clk.<br>01b - Select AON_CLK as watchdog clk.<br>10b - Select FOSC as watchdog clk.<br>11b - Select SIRC as watchdog clk. |
| **7**<br>**ENABLE** | RW | Watchdog Enable<br>Enable the watchdog counter. This bit is writeable once after reset.<br>0b - Watchdog disabled.<br>1b - Watchdog enabled. |
| **6**<br>**INT** | RW | Watchdog Interrupt<br>Enable the Watchdog interrupt. After setting this bit, the watchdog will generate an interrupt request once a reset-triggering event occurs before forcing a reset. The reset-triggering events include counter timeout or illegal writing to the watchdog. After the interrupt vector being fetched, the reset will be generated after 512 bus clock cycles by default. The number of bus clock cycles is configurable via CS[DLY_CNT_MSB]. This bit is writeable once after reset.<br>0b - Watchdog interrupts disabled. Watchdog resets are not delayed.<br>1b - Watchdog interrupts enabled. Watchdog resets are delayed by 512 bus clock cycles by default from the interrupt vector fetched. |
| **5**<br>**UPDATE** | RW | Allow Updates<br>Software is allowed to reconfigure the watchdog without a reset. This bit is writeable once after reset.<br>0b - Update not allowed. If there is not a forced reset, the watchdog cannot be reconfigured after initialization.<br>1b - Update allowed. The watchdog configuration registers can be modified by software within 128 bus clocks after operating the right unlock sequence. |

| | | |
|---|---|---|
| **4-3**<br>**TST** | RW | Watchdog Self Check<br>Enables the fast mode. In self-check mode, software will be allowed to toggle all bits of the counter to check whether there are any functional defects. These bits are writeable once and can only be cleared on POR.<br>00b - Watchdog self-check mode disabled.<br>01b - Watchdog user mode enabled. In this mode, the watchdog self-check mode is still disabled. After checking the watchdog, software should use this setting to indicate that the watchdog is functionally normal in user mode.<br>10b - Watchdog self-check mode enabled, only the low byte is used. COUNTER[CNTL] is compared with TIMEOUT[VALL].<br>11b - Watchdog self-check mode enabled, only the high byte is used. COUNTER[CNTH] is compared with TIMEOUT[VALH]. |
| **2**<br>**DBG** | RW | Debug Enable<br>Enable the watchdog to run when the chip is in debug mode. This bit is writeable once after reset.<br>0b - Watchdog disabled in chip debug mode.<br>1b - Watchdog enabled in chip debug mode. |
| **1**<br>**WAIT** | RW | Wait Enable<br>Enable the watchdog to run when the chip is in wait mode. This bit is writeable once after reset.<br>0b - Watchdog disabled in chip wait mode.<br>1b - Watchdog enabled in chip wait mode. |
| **0**<br>**STOP** | RW | Stop Enable<br>Enable the watchdog to run when the chip is in stop mode. This bit is writeable once after reset.<br>0b - Watchdog disabled in chip stop mode.<br>1b - Watchdog enabled in chip stop mode. |

### 33.8.2 WDOG_COUNTER (Watchdog Counter Register)

This section includes the description of the Watchdog Counter Register. The counter register is composed of a high 8-bit counter and a low 8-bit counter.

The COUNTER register will store the value of the watchdog counter, and provides the access for software to get the value. Writing to the counter register directly is illegal and will force a reset. The right methods are as follows:

1. The refresh sequence resets the watchdog counter to 0x0000.
2. The unlock sequence allows the watchdog to be reconfigured without forcing a reset (when CS[UPDATE] = 1).

Offset : 4h

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

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CNTH | | | | | | | | CNTL | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**WDOG_COUNTER field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-8<br>CNTH** | RW | High 8 bit of the watchdog counter |
| **7-0<br>CNTL** | RW | Low 8 bit of the watchdog counter |

### 33.8.3 WDOG_TIMEOUT (Watchdog Timeout Value Register)

This section includes the description of the Watchdog Timeout Value Register. The TIMEOUT register is 16-bit and is used to set the timeout range of the watchdog. This register is writeable once after reset.

The watchdog counter (COUNTER) is compared with the timeout value (TIMEOUT) continuously. If the counter reaches the timeout value, the watchdog forces a reset triggering event.

Do not write 0 to the TIMEOUT register (if CS[TST]=11b, then TIMEOUT_VALH cannot be written as 0; if CS[TST]=10b, then TIMEOUT_VALL cannot be 0); otherwise, the watchdog always generates a reset.

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TIMEOUT_VALH | | | | | | | | TIMEOUT_VALL | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**WDOG_TIMEOUT field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **15-8<br>TIMEOUT_VALH** | RW | High 8 bit of the watchdog timeout value |
| **7-0<br>TIMEOUT_VALL** | RW | Low 8 bit of the watchdog timeout value |

### 33.8.4 WDOG_WINDOW (Watchdog Window Register)

This section includes the description of the Watchdog Window Register. When window mode is enabled (CS[WIN] is set), the WINDOW register determines the earliest time that a refresh sequence is considered valid. This register is writeable once after reset.

The WINDOW register value must be less than the TIMEOUT register value. Otherwise, the watchdog will never be refreshed before timeout.

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WINDOW_HIGH | | | | | | | | WINDOW_LOW | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**WDOG_WINDOW field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-8<br>WINDOW_HIGH** | RW | High 8 bit of the watchdog window |
| **7-0<br>WINDOW_LOW** | RW | Low 8 bit of the watchdog window |

# Chapter 34 Error Injection Module (EIM)

## 34.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | Added support for single-bit or all-bits error injection to monitors to Section *Features* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 34.2 Chip-specific Information

N/A

## 34.3 Features

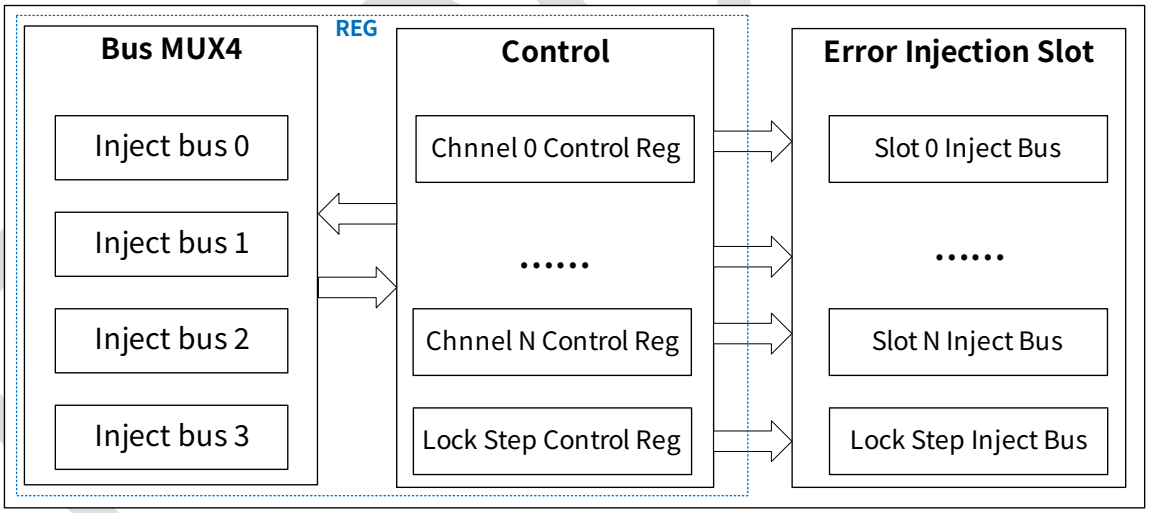
- Support several channels to inject single- or double-bit data, checkbit errors, and single-bit address errors to the corresponding memory
- Support several channels to inject single- or double-bit data, checkbit errors, single-bit address, and attribute errors (attribute indicates AHB bus interface signals) to the matrix
- Support several channels to inject single-bit errors or all-bits errors to the downsize monitor, AHBP/AHBS monitor, AFCB monitor, and CPU overlay
- Support CPU0 lockstep error injection
- Support DMA0 lockstep error injection
- Protection against accidental enable and reconfiguration error injection function via a two-stage enable mechanism

## 34.4 Overview

The Error Injection Module (EIM) is mainly used for Memory ECC logic and Matrix ECC logic self-test and diagnostic purposes. It provides a method for diagnostic coverage of the peripheral memories and matrix. Each EIM channel corresponds to a specific memory or matrix.

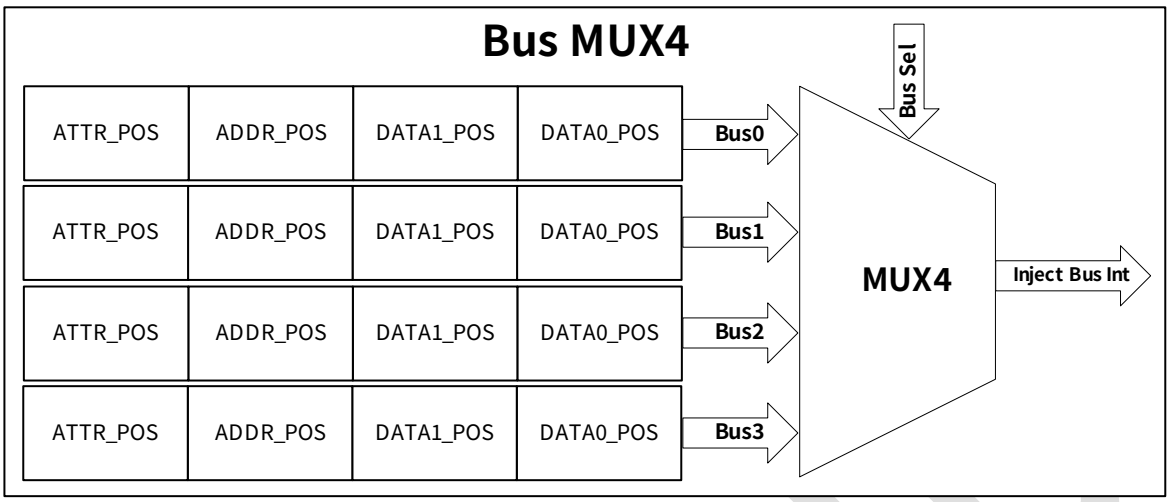
Below is the block diagram of EIM.

**Figure 46. EIM block diagram**



The EIM has four bus registers and mux4.

**Figure 47. EIM Bus MUX4 diagram**




## 34.5 Functional Description

EIM register access mode:

- Only in supervisor mode with 32-bit access can EIM registers be programmed
- In user mode, or for non-32-bit access or undefined address access, the EIM would generate hard fault interrupts

EIM error injection steps:

- For Memory and Matrix:
  1) Configure EIM_BUS_REG to select data0, data1, address, and attribute error injection position.
  2) Configure EIM_CTRL_REGn BUS_SEL bit to select EIM_BUS_REG.
  3) Configure EIM_CR to enable the error injection function of the EIM module.
  4) Configure EIM_CTRL_REGn to enable the data0, data1, address, and attribute error injection.
  5) Read from the memory or matrix slave being injected to see the error.
  6) Enter the error handling routine.
  7) Configure EIM_CTRL_REGn to disable the data0, data1, address, and attribute error injection.
  8) Configure EIM_CR to disable the error injection function of the EIM module.
  9) Configure EIM_BUS_REG to clear the error injection position.
- For CPU Lockstep:

  Taking the CPU0 lockstep error injection as an example:

  1) Configure LOCK STEP _MONx_SET of the CPU0_LOCKSTEP register.
  2) CPU0 accesses any slave.
  3) Enter the error handling routine (FCSMU).
  4) Configure LOCKSTEP_MONx_CLR of the CPU0_LOCKSTEP register.

## 34.6 Register Definition

EIM base address : 4001_9000h

**Register Overview - EIM**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | EIM_CR (Configuration Register) | 32 | RW | 0000_0000h | page |
| **10h+n*4h** | EIM_CTRL_REGn (Channel N Control Register) | 32 | RW | 0000_0000h | page |
| **400h** | EIM_CPU0_LOCKSTEP (CPU0 LOCKSTEP Error Injection Register) | 32 | RW | 0000_0000h | page |
| **408h** | EIM_DMA0_LOCKSTEP (DMA0 LOCKSTEP Error Injection Register) | 32 | RW | 0000_0000h | page |
| **800h+n*4h** | EIM_BUS_REGn (Bus Register) | 32 | RW | 0000_0000h | page |

### 34.6.1 EIM_CR (Configuration Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | GEIEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EIM_CR field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**GEIEN** | RW | Global Error Injection Enable<br>0b - Disable the global error injection function of the EIM<br>1b - Enable the global error injection function of the EIM |

### 34.6.2 EIM_CTRL_REGn (Channel N Control Register)

Offset : 10h+N*4h (N : 0 ~ 82)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DWP_LOCK | DWP | | | 0 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | BUS_SEL | | ATTREIE | ADDREIE | DATA1EIE | DATA0EIE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

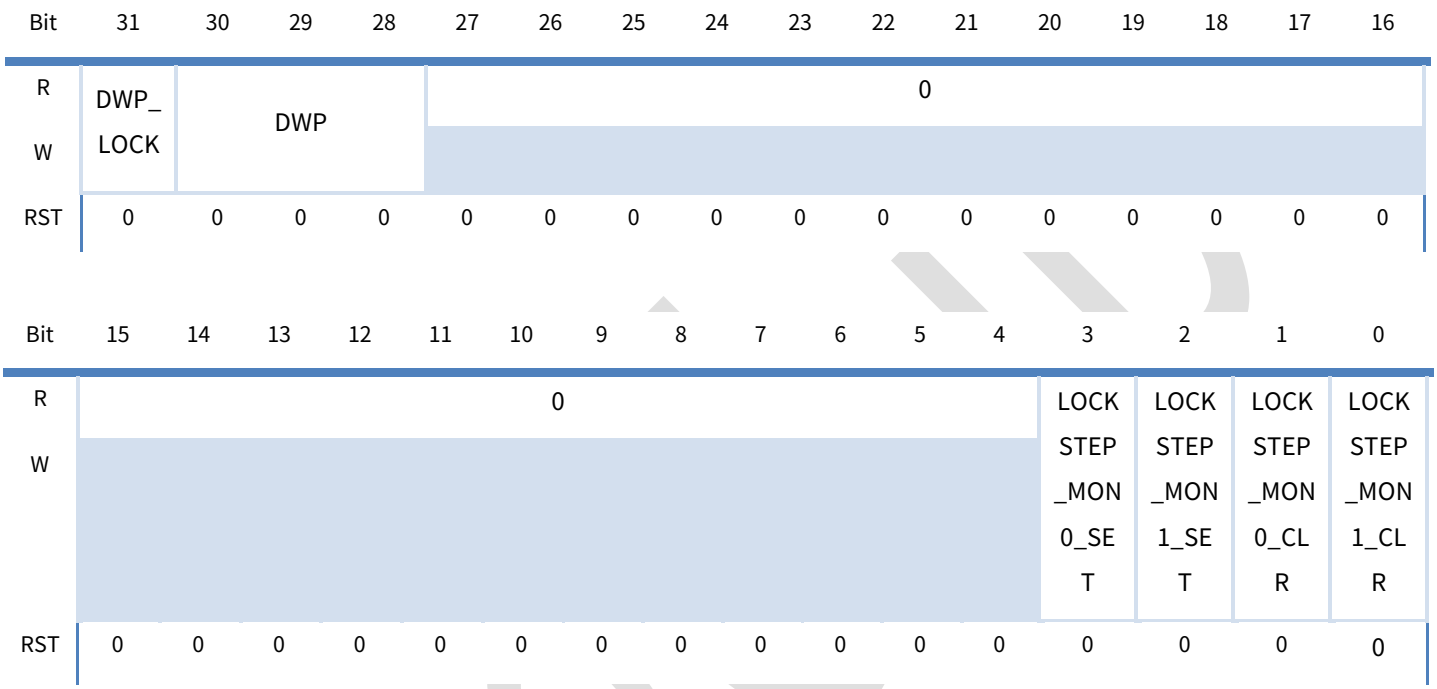
**EIM_CTRL_REGn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**DWP_LOCK** | RW | Domain Write Protection Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**DWP** | RW | Domain Write Protection<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |
| **5-4**<br>**BUS_SEL** | RW | Error Injection Bus Select<br>00b - Select BUS REG0 to injection<br>01b - Select BUS REG1 to injection<br>10b - Select BUS REG2 to injection<br>11b - Select BUS REG3 to injection |
| **3**<br>**ATTREIE** | RW | Channel N attribute error injection with CR[GEIEN] field must also be asserted.<br>0b - Channel N attribute error injection disable<br>1b - Channel N attribute error injection enable |
| **2**<br>**ADDREIE** | RW | Channel N address error injection with CR[GEIEN] field must also be asserted.<br>0b - Channel N address error injection disable<br>1b - Channel N address error injection enable |
| **1**<br>**DATA1EIE** | RW | Channel N data1 error injection with CR[GEIEN] field must also be asserted.<br>0b - Channel N data1 error injection disable<br>1b - Channel N data1 error injection enable |

| | | |
|---|---|---|
| **0**<br>**DATA0EIE** | RW | Channel N data0 error injection with CR[GEIEN] field must also be asserted.<br>0b - Channel N data0 error injection disable<br>1b - Channel N data0 error injection enable |

### 34.6.3 EIM_CPU0_LOCKSTEP (CPU0 LOCKSTEP Error Injection Register)

Offset : 400h




**EIM_CPU0_LOCKSTEP field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**DWP_LOCK** | RW | Domain Write Protection Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**DWP** | RW | Domain Write Protection<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |
| **3**<br>**LOCKSTEP_MON0_SET** | RW | CPU0 Lockstep Monitor0 Error Injection Set<br>0b - Not set<br>1b - Set |

| | | |
|---|---|---|
| **2**<br>**LOCKSTEP_MON1_SET** | RW | CPU0 Lockstep Monitor1 Error Injection Set<br>0b - Not set<br>1b - Set |
| **1**<br>**LOCKSTEP_MON0_CLR** | RW | CPU0 Lockstep Monitor0 Error Clear<br>0b - Not clear<br>1b - Clear |
| **0**<br>**LOCKSTEP_MON1_CLR** | RW | CPU0 Lockstep Monitor1 Error Clear<br>0b - Not clear<br>1b - Clear |

### 34.6.4 EIM_DMA0_LOCKSTEP (DMA0 LOCKSTEP Error Injection Register)

Offset : 408h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DWP_LOCK | DWP | | | 0 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | LOCKSTEP_MON0_SET | LOCKSTEP_MON1_SET | LOCKSTEP_MON0_CLR | LOCKSTEP_MON1_CLR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EIM_DMA0_LOCKSTEP field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**DWP_LOCK** | RW | Domain Write Protection Lock<br>0b - Not locked<br>1b - Locked |
| **30-28**<br>**DWP** | RW | Domain Write Protection<br>This field indicates which core is allowed to access this bit.<br>000b - CPU0 is allowed to access.<br>001b - CPU0 is allowed to access.<br>010b - Reserved<br>011b - Reserved<br>Others - Non-CPU masters are allowed to access. |

| | | |
|---|---|---|
| **3**<br>**LOCKSTEP_MON0_SET** | RW | DMA0 Lockstep Monitor0 Error Injection Set<br>0b - Not set<br>1b - Set |
| **2**<br>**LOCKSTEP_MON1_SET** | RW | DMA0 Lockstep Monitor1 Error Injection Set<br>0b - Not set<br>1b - Set |
| **1**<br>**LOCKSTEP_MON0_CLR** | RW | DMA0 Lockstep Monitor0 Error Clear<br>0b - Not clear<br>1b - Clear |
| **0**<br>**LOCKSTEP_MON1_CLR** | RW | DMA0 Lockstep Monitor1 Error Clear<br>0b - Not clear<br>1b - Clear |

### 34.6.5 EIM_BUS_REGn (Bus Register)

Offset : 800h+N*4h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | ATTR | | | | | 0 | | | ADDR | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DATA1 | | | | | | | 0 | DATA0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**EIM_BUS_REGn field descriptions**

| Field | Type | Description |
|---|---|---|
| **28-24**<br>**ATTR** | RW | Attribute Single Bit Error Injection Position<br>For a single-bit attribute error, configuring EIM_BUS_REG[ATTR] = 6 means bit 6 of the attribute inject error. Value 0~25 are allowed. |
| **20-16**<br>**ADDR** | RW | Address Single Bit Error Injection Position<br>For a single-bit address error, configuring EIM_BUS_REG[ADDR] = 4 means bit 4 of the address inject error. Value 3~30 are allowed. |

| | | |
|---|---|---|
| **14-8**<br>**DATA1** | RW | Data1 Single Bit Error Injection Position<br>For a single-bit data error, configuring EIM_BUS_REG[DATA1] = 3 means bit 3 of the data inject error. Value 0~71 are allowed.<br>For a double-bit data error, configuring EIM_BUS_REG[DATA0] = 5 and EIM_BUS_REG[DATA1] = 7 means bit 5 and bit 7 of the data inject error. Value 0~71 are allowed. |
| **6-0**<br>**DATA0** | RW | Data0 Single Bit Error Injection Position<br>For a single-bit data error, configuring EIM_BUS_REG[DATA0] = 3 means bit 3 of the data inject error. Value 0~71 are allowed.<br>For a double-bit data error, configuring EIM_BUS_REG[DATA0] = 5 and EIM_BUS_REG[DATA1] = 7 means bit 5 and bit 7 of the data inject error. Value 0~71 are allowed. |

Flagchip

# Chapter 35 Error Reporting Module (ERM)

## 35.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 35.2 Chip-specific Information

N/A

## 35.3 Features

- Support 28 channels for reporting corresponding memory error event source
- Support reporting the address information on single-bit correction and non-correctable Error Correction Code (ECC) events
- Selectable interrupt request assertion on captured ECC events
- Support for ECC event capturing for memory sources, with individual reporting fields and interrupt configuration per memory channel

## 35.4 Overview

The Error Reporting Module (ERM) provides information and optional interrupt notification on memory error events associated with ECC. The ERM collects ECC events on memory accesses for memory arrays, such as flash memory, CPU RAM, system RAM, ROM or DMA RAMs.

## 35.5 Functional Description

ERM register access mode:

- Only in supervisor mode with 32-bit access can ERM registers be programmed
- In user mode or for non-32-bit access, the ERM would generate hard fault interrupts
- Access to an undefined address can result in undefined behavior

### 35.5.1 Initialization

For each ERM channel, prepare the corresponding memory array before enabling ERM interrupts about errors for that memory.

- Initialize the memory to a known value so that the correct corresponding ECC codeword is stored.
- During the memory's initialization, if the ERM captures information about any ECC error event, clear the corresponding SRx[SBCn] or SRx[NCEn] field that stores the record of the event.
- Program the applicable CRx[ESCIEn] and CRx[ENCIEn] fields to enable ERM interrupts as desired.

### 35.5.2 Single-bit Correction Events

The ERM holds event information only for the last reported event. When a single-bit correction event on Memory *n* is detected, the ERM:

- Records the event by changing the value of the applicable Status Register bit SRx[SBCn] to 1, and write 1 to SRx[SBCn] to change its value to 0 to clear the record of an event.
- Records the corresponding access address that initiated the event in the Memory n Error Address Register: EARn.

The ERM provides a selectable interrupt notification upon the report of a single-bit correction event:

- Set CRx[ESCIEn] to enable interrupt notification for single-bit correction events on Memory n.
- When a single-bit correction event on Memory n is detected, the ERM: records the event and address, and sends an interrupt notification corresponding to the event.
- Write 1 to SRx[SBCn] to clear both the record of an event and the corresponding interrupt notification.

### 35.5.3 Non-correctable Error Events

The ERM holds event information only for the last reported event. When a non-correctable ECC error event on Memory *n* is detected, the ERM:

- Records the event by changing the value of the applicable Status Register bit: SRx[NCEn]to 1, and write 1 to SRx[NCEn] to change its value to 0 to clear the record of an event.
- Records the corresponding access address that initiated the event in the Memory n Error Address Register: EARn.

The ERM provides a selectable interrupt notification upon the report of a non-correctable ECC event:

- Set CRx[ENCIEn] to enable interrupt notifications for non-correctable error events on Memory n.
- When a non-correctable error event on Memory n is detected, the ERM: records the event and address and sends an interrupt notification corresponding to the event.
- Write 1 to SRx[NCEn] to clear both the record of an event and the corresponding interrupt notification.

## 35.6 Register Definition

ERM base address : 4001_8000h

**Register Overview - ERM**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | ERM_CR0 (Configuration Register 0) | 32 | RW | 0000_0000h | page |
| **4h** | ERM_CR1 (Configuration Register 1) | 32 | RW | 0000_0000h | page |
| **8h** | ERM_CR2 (Configuration Register 2) | 32 | RW | 0000_0000h | page |
| **Ch** | ERM_CR3 (Configuration Register 3) | 32 | RW | 0000_0000h | page |
| **10h** | ERM_SR0 (Status Register 0) | 32 | W1C | 0000_0000h | page |
| **14h** | ERM_SR1 (Status Register 1) | 32 | W1C | 0000_0000h | page |
| **18h** | ERM_SR2 (Status Register 2) | 32 | W1C | 0000_0000h | page |
| **1Ch** | ERM_SR3 (Status Register 3) | 32 | W1C | 0000_0000h | page |
| **100h+n*10h** | ERM_EARn (Error Address Register) | 32 | RO | 0000_0000h | page |

### 35.6.1 ERM_CR0 (Configuration Register 0)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ESCIE0 | ENCIE0 | 0 | | ESCIE1 | ENCIE1 | 0 | | ESCIE2 | ENCIE2 | 0 | | ESCIE3 | ENCIE3 | 0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ESCIE4 | ENCIE4 | 0 | | ESCIE5 | ENCIE5 | 0 | | ESCIE6 | ENCIE6 | 0 | | ESCIE7 | ENCIE7 | 0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ERM_CR0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**ESCIE0** | RW | Enable Memory 0 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 0 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 0 single-bit correction events. |
| **30**<br>**ENCIE0** | RW | Enable Memory 0 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 0 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 0 non-correctable error events. |
| **27**<br>**ESCIE1** | RW | Enable Memory 1 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 1 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 1 single-bit correction events. |
| **26**<br>**ENCIE1** | RW | Enable Memory 1 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 1 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 1 non-correctable error events. |
| **23**<br>**ESCIE2** | RW | Enable Memory 2 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 2 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 2 single-bit correction events. |
| **22**<br>**ENCIE2** | RW | Enable Memory 2 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 2 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 2 non-correctable error events. |
| **19**<br>**ESCIE3** | RW | Enable Memory 3 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 3 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 3 single-bit correction events. |
| **18**<br>**ENCIE3** | RW | Enable Memory 3 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 3 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 3 non-correctable error events. |
| **15**<br>**ESCIE4** | RW | Enable Memory 4 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 4 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 4 single-bit correction events. |
| **14**<br>**ENCIE4** | RW | Enable Memory 4 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 4 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 4 non-correctable error events. |

| | | |
|---|---|---|
| **11**<br>**ESCIE5** | RW | Enable Memory 5 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 5 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 5 single-bit correction events. |
| **10**<br>**ENCIE5** | RW | Enable Memory 5 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 2 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 2 non-correctable error events. |
| **7**<br>**ESCIE6** | RW | Enable Memory 6 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 6 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 6 single-bit correction events. |
| **6**<br>**ENCIE6** | RW | Enable Memory 6 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 6 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 6 non-correctable error events. |
| **3**<br>**ESCIE7** | RW | Enable Memory 7 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 7 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 7 single-bit correction events. |
| **2**<br>**ENCIE7** | RW | Enable Memory 7 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 7 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 7 non-correctable error events. |

### 35.6.2 ERM_CR1 (Configuration Register 1)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ESCIE8 | ENCIE8 | 0 | | ESCIE9 | ENCIE9 | 0 | | ESCIE10 | ENCIE10 | 0 | | ESCIE11 | ENCIE11 | 0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ESCIE12 | ENCIE12 | 0 | | ESCIE13 | ENCIE13 | 0 | | ESCIE14 | ENCIE14 | 0 | | ESCIE15 | ENCIE15 | 0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ERM_CR1 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31**<br>**ESCIE8** | RW | Enable Memory 8 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 8 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 8 single-bit correction events. |
| **30**<br>**ENCIE8** | RW | Enable Memory 8 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 8 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 8 non-correctable error events. |
| **27**<br>**ESCIE9** | RW | Enable Memory 9 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 9 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 9 single-bit correction events. |
| **26**<br>**ENCIE9** | RW | Enable Memory 9 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 9 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 9 non-correctable error events. |
| **23**<br>**ESCIE10** | RW | Enable Memory 10 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 10 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 10 single-bit correction events. |
| **22**<br>**ENCIE10** | RW | Enable Memory 10 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 10 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 10 non-correctable error events. |
| **19**<br>**ESCIE11** | RW | Enable Memory 11 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 11 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 11 single-bit correction events. |
| **18**<br>**ENCIE11** | RW | Enable Memory 11 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 11 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 11 non-correctable error events. |
| **15**<br>**ESCIE12** | RW | Enable Memory 12 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 12 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 12 single-bit correction events. |
| **14**<br>**ENCIE12** | RW | Enable Memory 12 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 12 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 12 non-correctable error events. |

| | | |
|---|---|---|
| **11**<br>**ESCIE13** | RW | Enable Memory 13 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 13 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 13 single-bit correction events. |
| **10**<br>**ENCIE13** | RW | Enable Memory 13 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 13 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 13 non-correctable error events. |
| **7**<br>**ESCIE14** | RW | Enable Memory 14 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 14 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 14 single-bit correction events. |
| **6**<br>**ENCIE14** | RW | Enable Memory 14 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 14 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 14 non-correctable error events. |
| **3**<br>**ESCIE15** | RW | Enable Memory 15 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 15 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 15 single-bit correction events. |
| **2**<br>**ENCIE15** | RW | Enable Memory 15 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 15 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 15 non-correctable error events. |

### 35.6.3 ERM_CR2 (Configuration Register 2)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ESCIE16 | ENCIE16 | 0 | | ESCIE17 | ENCIE17 | 0 | | ESCIE18 | ENCIE18 | 0 | | ESCIE19 | ENCIE19 | 0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ESCIE20 | ENCIE20 | 0 | | ESCIE21 | ENCIE21 | 0 | | ESCIE22 | ENCIE22 | 0 | | ESCIE23 | ENCIE23 | 0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ERM_CR2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**ESCIE16** | RW | Enable Memory 16 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 16 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 16 single-bit correction events. |
| **30**<br>**ENCIE16** | RW | Enable Memory 16 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 16 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 16 non-correctable error events. |
| **27**<br>**ESCIE17** | RW | Enable Memory 17 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 17 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 17 single-bit correction events. |
| **26**<br>**ENCIE17** | RW | Enable Memory 17 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 17 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 17 non-correctable error events. |
| **23**<br>**ESCIE18** | RW | Enable Memory 18 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 18 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 18 single-bit correction events. |
| **22**<br>**ENCIE18** | RW | Enable Memory 18 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 18 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 18 non-correctable error events. |
| **19**<br>**ESCIE19** | RW | Enable Memory 19 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 19 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 19 single-bit correction events. |
| **18**<br>**ENCIE19** | RW | Enable Memory 19 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 19 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 19 non-correctable error events. |
| **15**<br>**ESCIE20** | RW | Enable Memory 20 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 20 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 20 single-bit correction events. |

| | | |
|---|---|---|
| **14**<br>**ENCIE20** | RW | Enable Memory 20 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 20 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 20 non-correctable error events. |
| **11**<br>**ESCIE21** | RW | Enable Memory 21 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 21 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 21 single-bit correction events. |
| **10**<br>**ENCIE21** | RW | Enable Memory 21 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 21 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 21 non-correctable error events. |
| **7**<br>**ESCIE22** | RW | Enable Memory 22 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 22 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 22 single-bit correction events. |
| **6**<br>**ENCIE22** | RW | Enable Memory 22 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 22 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 22 non-correctable error events. |
| **3**<br>**ESCIE23** | RW | Enable Memory 23 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 23 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 23 single-bit correction events. |
| **2**<br>**ENCIE23** | RW | Enable Memory 23 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 23 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 23 non-correctable error events. |

### 35.6.4 ERM_CR3 (Configuration Register 3)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ESCIE24 | ENCIE24 | 0 | | ESCIE25 | ENCIE25 | 0 | | ESCIE26 | ENCIE26 | 0 | | ESCIE27 | ENCIE27 | 0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ESCI E28 | ENCI E28 | 0 | | ESCI E29 | ENCI E29 | 0 | | ESCI E30 | ENCI E30 | 0 | | ESCI E31 | ENCI E31 | 0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ERM_CR3 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**ESCIE24** | RW | Enable Memory 24 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 24 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 24 single-bit correction events. |
| **30**<br>**ENCIE24** | RW | Enable Memory 24 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 24 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 24 non-correctable error events. |
| **27**<br>**ESCIE25** | RW | Enable Memory 25 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 25 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 25 single-bit correction events. |
| **26**<br>**ENCIE25** | RW | Enable Memory 25 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 25 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 25 non-correctable error events. |
| **23**<br>**ESCIE26** | RW | Enable Memory 26 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 26 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 26 single-bit correction events. |
| **22**<br>**ENCIE26** | RW | Enable Memory 26 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 26 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 26 non-correctable error events. |
| **19**<br>**ESCIE27** | RW | Enable Memory 27 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 27 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 27 single-bit correction events. |
| **18**<br>**ENCIE27** | RW | Enable Memory 27 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 27 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 27 non-correctable error events. |

| | | |
|---|---|---|
| **15**<br>**ESCIE28** | RW | Enable Memory 28 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 28 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 28 single-bit correction events. |
| **14**<br>**ENCIE28** | RW | Enable Memory 28 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 28 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 28 non-correctable error events. |
| **11**<br>**ESCIE29** | RW | Enable Memory 29 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 29 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 29 single-bit correction events. |
| **10**<br>**ENCIE29** | RW | Enable Memory 29 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 29 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 29 non-correctable error events. |
| **7**<br>**ESCIE30** | RW | Enable Memory 30 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 30 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 30 single-bit correction events. |
| **6**<br>**ENCIE30** | RW | Enable Memory 30 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 30 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 30 non-correctable error events. |
| **3**<br>**ESCIE31** | RW | Enable Memory 31 Single Correction Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 31 single-bit correction events.<br>1b - Enable interrupt reporting of Memory 31 single-bit correction events. |
| **2**<br>**ENCIE31** | RW | Enable Memory 31 Non-Correctable Interrupt Reporting<br>0b - Disable interrupt reporting of Memory 31 non-correctable error events.<br>1b - Enable interrupt reporting of Memory 31 non-correctable error events. |

### 35.6.5 ERM_SR0 (Status Register 0)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SBC0 | NCE0 | 0 | | SBC1 | NCE1 | 0 | | SBC2 | NCE2 | 0 | | SBC3 | NCE3 | 0 | |
| W | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SBC4 | NCE4 | 0 | | SBC5 | NCE5 | 0 | | SBC6 | NCE6 | 0 | | SBC7 | NCE7 | 0 | |
| W | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ERM_SR0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**SBC0** | W1C | Memory 0 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 0 detected.<br>1b - Single-bit correction event on Memory 0 detected. |
| **30**<br>**NCE0** | W1C | Memory 0 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 0 detected.<br>1b - Non-correctable error event on Memory 0 detected. |
| **27**<br>**SBC1** | W1C | Memory 1 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 1 detected.<br>1b - Single-bit correction event on Memory 1 detected. |
| **26**<br>**NCE1** | W1C | Memory 1 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 1 detected.<br>1b - Non-correctable error event on Memory 1 detected. |
| **23**<br>**SBC2** | W1C | Memory 2 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 2 detected.<br>1b - Single-bit correction event on Memory 2 detected. |
| **22**<br>**NCE2** | W1C | Memory 2 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 2 detected.<br>1b - Non-correctable error event on Memory 2 detected. |

| | | |
|---|---|---|
| **19**<br>**SBC3** | W1C | Memory 3 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 3 detected.<br>1b - Single-bit correction event on Memory 3 detected. |
| **18**<br>**NCE3** | W1C | Memory 3 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 3 detected.<br>1b - Non-correctable error event on Memory 3 detected. |
| **15**<br>**SBC4** | W1C | Memory 4 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 4 detected.<br>1b - Single-bit correction event on Memory 4 detected. |
| **14**<br>**NCE4** | W1C | Memory 4 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 4 detected.<br>1b - Non-correctable error event on Memory 4 detected. |
| **11**<br>**SBC5** | W1C | Memory 5 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 5 detected.<br>1b - Single-bit correction event on Memory 5 detected. |
| **10**<br>**NCE5** | W1C | Memory 5 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 5 detected.<br>1b - Non-correctable error event on Memory 5 detected. |
| **7**<br>**SBC6** | W1C | Memory 6 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 6 detected.<br>1b - Single-bit correction event on Memory 6 detected. |
| **6**<br>**NCE6** | W1C | Memory 6 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 6 detected.<br>1b - Non-correctable error event on Memory 6 detected. |

| | | |
|---|---|---|
| **3**<br>**SBC7** | W1C | Memory 7 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 7 detected.<br>1b - Single-bit correction event on Memory 7 detected. |
| **2**<br>**NCE7** | W1C | Memory 7 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 7 detected.<br>1b - Non-correctable error event on Memory 7 detected. |

### 35.6.6 ERM_SR1 (Status Register 1)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SBC8 | NCE8 | 0 | | SBC9 | NCE9 | 0 | | SBC10 | NCE10 | 0 | | SBC11 | NCE11 | 0 | |
| W | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SBC12 | NCE12 | 0 | | SBC13 | NCE13 | 0 | | SBC14 | NCE14 | 0 | | SBC15 | NCE15 | 0 | |
| W | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ERM_SR1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**SBC8** | W1C | Memory 8 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 8 detected.<br>1b - Single-bit correction event on Memory 8 detected. |
| **30**<br>**NCE8** | W1C | Memory 8 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 8 detected.<br>1b - Non-correctable error event on Memory 8 detected. |

| | | |
|---|---|---|
| **27**<br>**SBC9** | W1C | Memory 9 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 9 detected.<br>1b - Single-bit correction event on Memory 9 detected. |
| **26**<br>**NCE9** | W1C | Memory 9 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 9 detected.<br>1b - Non-correctable error event on Memory 9 detected. |
| **23**<br>**SBC10** | W1C | Memory 10 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 10 detected.<br>1b - Single-bit correction event on Memory 10 detected. |
| **22**<br>**NCE10** | W1C | Memory 10 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 10 detected.<br>1b - Non-correctable error event on Memory 10 detected. |
| **19**<br>**SBC11** | W1C | Memory 11 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 11 detected.<br>1b - Single-bit correction event on Memory 11 detected. |
| **18**<br>**NCE11** | W1C | Memory 11 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 11 detected.<br>1b - Non-correctable error event on Memory 11 detected. |
| **15**<br>**SBC12** | W1C | Memory 12 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 12 detected.<br>1b - Single-bit correction event on Memory 12 detected. |
| **14**<br>**NCE12** | W1C | Memory 12 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 12 detected.<br>1b - Non-correctable error event on Memory 12 detected. |

| | | |
|---|---|---|
| **11**<br>**SBC13** | W1C | Memory 13 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 13 detected.<br>1b - Single-bit correction event on Memory 13 detected. |
| **10**<br>**NCE13** | W1C | Memory 13 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 13 detected.<br>1b - Non-correctable error event on Memory 13 detected. |
| **7**<br>**SBC14** | W1C | Memory 14 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 14 detected.<br>1b - Single-bit correction event on Memory 14 detected. |
| **6**<br>**NCE14** | W1C | Memory 14 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 14 detected.<br>1b - Non-correctable error event on Memory 14 detected. |
| **3**<br>**SBC15** | W1C | Memory 15 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 15 detected.<br>1b - Single-bit correction event on Memory 15 detected. |
| **2**<br>**NCE15** | W1C | Memory 15 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 15 detected.<br>1b - Non-correctable error event on Memory 15 detected. |

### 35.6.7 ERM_SR2 (Status Register 2)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SBC16 | NCE16 | 0 | | SBC17 | NCE17 | 0 | | SBC18 | NCE18 | 0 | | SBC19 | NCE19 | 0 | |
| W | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SBC20 | NCE20 | 0 | | SBC21 | NCE21 | 0 | | SBC22 | NCE22 | 0 | | SBC23 | NCE23 | 0 | |
| W | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ERM_SR2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**SBC16** | W1C | Memory 16 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 16 detected.<br>1b - Single-bit correction event on Memory 16 detected. |
| **30**<br>**NCE16** | W1C | Memory 16 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 16 detected.<br>1b - Non-correctable error event on Memory 16 detected. |
| **27**<br>**SBC17** | W1C | Memory 17 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 17 detected.<br>1b - Single-bit correction event on Memory 17 detected. |
| **26**<br>**NCE17** | W1C | Memory 17 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 17 detected.<br>1b - Non-correctable error event on Memory 17 detected. |
| **23**<br>**SBC18** | W1C | Memory 18 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 18 detected.<br>1b - Single-bit correction event on Memory 18 detected. |
| **22**<br>**NCE18** | W1C | Memory 18 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 18 detected.<br>1b - Non-correctable error event on Memory 18 detected. |

| | | |
|---|---|---|
| **19**<br>**SBC19** | W1C | Memory 19 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 19 detected.<br>1b - Single-bit correction event on Memory 19 detected. |
| **18**<br>**NCE19** | W1C | Memory 19 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 19 detected.<br>1b - Non-correctable error event on Memory 19 detected. |
| **15**<br>**SBC20** | W1C | Memory 20 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 20 detected.<br>1b - Single-bit correction event on Memory 20 detected. |
| **14**<br>**NCE20** | W1C | Memory 20 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 20 detected.<br>1b - Non-correctable error event on Memory 20 detected. |
| **11**<br>**SBC21** | W1C | Memory 21 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 21 detected.<br>1b - Single-bit correction event on Memory 21 detected. |
| **10**<br>**NCE21** | W1C | Memory 21 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 21 detected.<br>1b - Non-correctable error event on Memory 21 detected. |
| **7**<br>**SBC22** | W1C | Memory 22 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 22 detected.<br>1b - Single-bit correction event on Memory 22 detected. |
| **6**<br>**NCE22** | W1C | Memory 22 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 22 detected.<br>1b - Non-correctable error event on Memory 22 detected. |

| | | |
|---|---|---|
| **3**<br>**SBC23** | W1C | Memory 23 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 23 detected.<br>1b - Single-bit correction event on Memory 23 detected. |
| **2**<br>**NCE23** | W1C | Memory 23 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 23 detected.<br>1b - Non-correctable error event on Memory 23 detected. |

### 35.6.8 ERM_SR3 (Status Register 3)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SBC24 | NCE24 | 0 | | SBC25 | NCE25 | 0 | | SBC26 | NCE26 | 0 | | SBC27 | NCE27 | 0 | |
| W | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SBC28 | NCE28 | 0 | | SBC29 | NCE29 | 0 | | SBC30 | NCE30 | 0 | | SBC31 | NCE31 | 0 | |
| W | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | | w1c | w1c | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ERM_SR3 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**SBC24** | W1C | Memory 24 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 24 detected.<br>1b - Single-bit correction event on Memory 24 detected. |
| **30**<br>**NCE24** | W1C | Memory 24 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 24 detected.<br>1b - Non-correctable error event on Memory 24 detected. |

| | | |
|---|---|---|
| **27**<br>**SBC25** | W1C | Memory 25 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 25 detected.<br>1b - Single-bit correction event on Memory 25 detected. |
| **26**<br>**NCE25** | W1C | Memory 25 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 25 detected.<br>1b - Non-correctable error event on Memory 25 detected. |
| **23**<br>**SBC26** | W1C | Memory 26 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 26 detected.<br>1b - Single-bit correction event on Memory 26 detected. |
| **22**<br>**NCE26** | W1C | Memory 26 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 26 detected.<br>1b - Non-correctable error event on Memory 26 detected. |
| **19**<br>**SBC27** | W1C | Memory 27 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 27 detected.<br>1b - Single-bit correction event on Memory 27 detected. |
| **18**<br>**NCE27** | W1C | Memory 27 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 27 detected.<br>1b - Non-correctable error event on Memory 27 detected. |
| **15**<br>**SBC28** | W1C | Memory 28 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 28 detected.<br>1b - Single-bit correction event on Memory 28 detected. |
| **14**<br>**NCE28** | W1C | Memory 28 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 28 detected.<br>1b - Non-correctable error event on Memory 28 detected. |

| | | |
|---|---|---|
| **11**<br>**SBC29** | W1C | Memory 29 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 29 detected.<br>1b - Single-bit correction event on Memory 29 detected. |
| **10**<br>**NCE29** | W1C | Memory 29 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 29 detected.<br>1b - Non-correctable error event on Memory 29 detected. |
| **7**<br>**SBC30** | W1C | Memory 30 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 30 detected.<br>1b - Single-bit correction event on Memory 30 detected. |
| **6**<br>**NCE30** | W1C | Memory 30 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 30 detected.<br>1b - Non-correctable error event on Memory 30 detected. |
| **3**<br>**SBC31** | W1C | Memory 31 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 31 detected.<br>1b - Single-bit correction event on Memory 31 detected. |
| **2**<br>**NCE31** | W1C | Memory 31 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt Reporting, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 31 detected.<br>1b - Non-correctable error event on Memory 31 detected. |

### 35.6.9 ERM_EARn (Error Address Register)

Offset : 100h+N*10h (N : 0 ~ 27)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EAR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EAR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ERM_EARn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**EAR** | RO | Memory n Error Address<br>This field contains the faulting system address of the last recorded ECC event on Memory n. |

Flagchip

# Chapter 36 Interrupt Monitor (INTM)

## 36.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • Changed the number of supported programmable interrupt monitors from "2" to "4" in Section *Features;*<br>• Changed the N range from "0-1" to "0-3" in the register sections *INTM_IRQSELRn, INTM_LATRn, INTM_TMRn,* and *INTM_SRn* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 36.2 Chip-specific Information

N/A

## 36.3 Features

- Support 4 programmable interrupt monitors with an optional interrupt or reset
- Selectable interrupt source per monitor
- Support 2 interrupt monitor modes: active mode and inactive mode
- Programmable 24-bit latency counter per monitor.
- Timer value exceeded latency value status bit per monitor
- One interrupt acknowledges for all monitors to stop counter

## 36.4 Overview

The Interrupt Monitor (INTM) module is used to monitor the latency of interrupt sources to ensure that these interrupt requests are executed within the expected time, increasing the reliability of the device.

## 36.5 Functional Description

### 36.5.1 INTM Register Access Mode

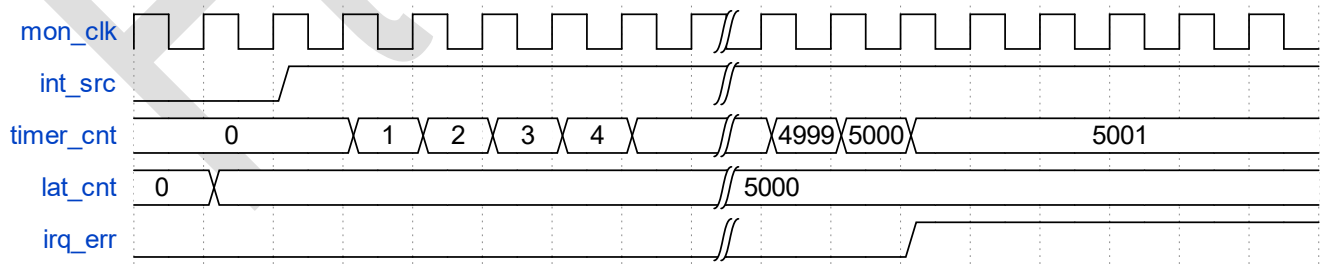
- Only in supervisor mode with 32-bit access can INTM registers be programmed.
- In user mode, or for non-32-bit access or undefined address access, the INTM will generate hard fault interrupts.

### 36.5.2 INTM Interrupt Active Mode

To configure INTM as interrupt active mode, perform the following steps:

1) Select an interrupt request source to be monitored by configuring INTM_IRQSELR[IRQ].
2) Select the INTM monitor mode by configuring INTM_IRQSELR[IACTE].
3) Program the INTM_IRQSELR[INTE] or INTM_IRQSELR[RSTE] to enable the interrupt or reset generation on timeout.
4) Select the expected latency time for the monitored interrupt by configuring INTM_LATR.
5) Enable INTM_ER to monitor the selected interrupt sources.
6) Write to INTM_IACKR when entering the interrupt service routine.
7) INTM_SR is set when INTM_TMR exceeds INTM_LATR.

**Figure 48. INTM interrupt active mode waveform**



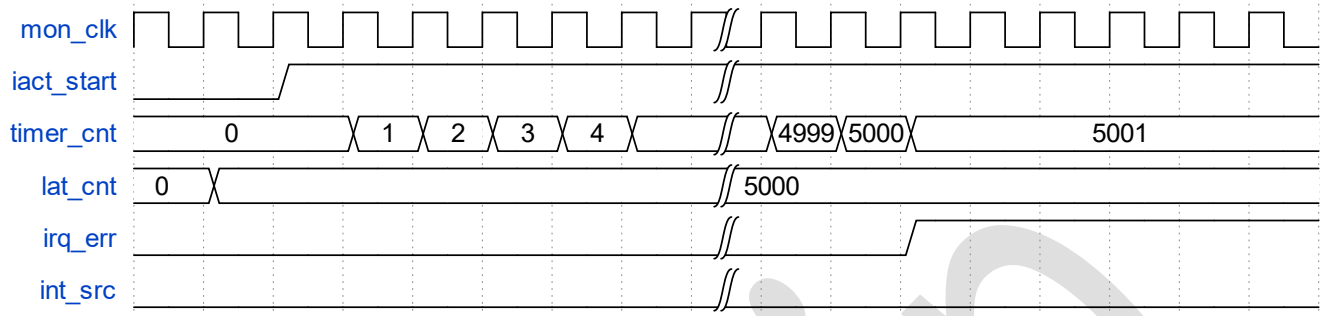

### 36.5.3 INTM Interrupt Inactive Mode

To configure INTM as interrupt inactive mode, perform the following steps:

1) Select an interrupt request source to be monitored by configuring INTM_IRQSELR[IRQ].
2) Select the INTM monitor mode by configuring INTM_IRQSELR[IACTE].
3) Program the INTM_IRQSELR[INTE] or INTM_IRQSELR[RSTE] to enable the interrupt or reset generation on timeout.
4) Select the expected latency time for the monitored interrupt by configuring INTM_LATR.
5) Set INTM_ER to enable the monitor.

6) Set INTM_IRQSELR[IACTST] to start the interrupt inactive counter.
7) Clear the timer counter and INTM_IRQSELR[IACTST] when entering the interrupt service routine.
8) INTM_SR is set when INTM_TMR exceeds INTM_LATR.

**Figure 49. INTM interrupt inactive mode waveform**




## 36.6 Register Definition

INTM base address : 4001_A000h

**Register Overview - INTM**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | INTM_ER (Enable Register) | 32 | RW | 0000_0000h | page |
| **4h** | INTM_IACKR (Interrupt Acknowledge Register) | 32 | RW | 0000_0000h | page |
| **8h+n*10h** | INTM_IRQSELRn (Interrupt Request Select Register) | 32 | RW | 0000_0000h | page |
| **Ch+n*10h** | INTM_LATRn (Latency Register) | 32 | RW | 0000_0000h | page |
| **10h+n*10h** | INTM_TMRn (Timer Register) | 32 | RW | 0000_0000h | page |
| **14h+n*10h** | INTM_SRn (Status Register) | 32 | RO | 0000_0000h | page |

### 36.6.1 INTM_ER (Enable Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**INTM_ER field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**EN** | RW | INTM Monitors Enable<br>0b - Disable INTM monitors<br>1b - Enable INTM monitors |

### 36.6.2 INTM_IACKR (Interrupt Acknowledge Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | 0 | | | | | | | | | |
| W | | | | | | | IRQ | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**INTM_IACKR field descriptions**

| Field | Type | Description |
|---|---|---|
| **9-0**<br>**IRQ** | WO | Select the interrupt request number to stop INTM_TIMERa increment. |

### 36.6.3 INTM_IRQSELRn (Interrupt Request Select Register)

Offset : 8h+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RSTE | INTE | IACTE | 0 | | | | | | | | | | | | IACTST |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | IRQ | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**INTM_IRQSELRn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**RSTE** | RW | Reset Enable<br>0b - Disable INTM monitors reset assert when INTM_TMR has exceeded the INTM_LATR value.<br>1b - Enable INTM monitors reset assert when INTM_TMR has exceeded the INTM_LATR value. |
| **30**<br>**INTE** | RW | Interrupt Enable<br>0b - Disable INTM monitors interrupt request when INTM_TMR has exceeded the INTM_LATR value.<br>1b - Enable INTM monitors interrupt request when INTM_TMR has exceeded the INTM_LATR value. |
| **29**<br>**IACTE** | RW | Inactive Mode Enable<br>0b - INTM is configured as active mode<br>1b - INTM is configured as inactive mode |
| **16**<br>**IACTST** | RW | Inactive Mode Start<br>This bit is used to start the interrupt inactive counter. It needs to be cleared manually.<br>0b - Inactive mode does not start<br>1b - Inactive mode starts |
| **9-0**<br>**IRQ** | RW | Interrupt Request Number<br>Select the interrupt request number to monitor. |

### 36.6.4 INTM_LATRn (Latency Register)

Offset : Ch+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | LAT | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LAT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**INTM_LATRn field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-0 LAT** | RW | Select the maximum number of INTM clock cycles allowed for the monitored interrupt request. |

### 36.6.5 INTM_TMRn (Timer Register)

Offset : 10h+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TIMER | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIMER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**INTM_TMRn field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-0 TIMER** | RW | Count the number of INTM clock cycles. |

### 36.6.6 INTM_SRn (Status Register)

Offset : 14h+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | STATUS |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**INTM_SRn field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**STATUS** | RO | Monitor Status<br>0b - INTM_TMR value has not exceeded the INTM_LATR value.<br>1b - INTM_TMR value has exceeded the INTM_LATR value. |

Flagchip

# Chapter 37 Clock Monitor Unit (CMU)

## 37.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | Added a note for confirming interrupt flags are cleared to Section *Chip-specific Information* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 37.2 Chip-specific Information

The FC7240 series has 5 instances for the Clock Monitor Unit (CMU). The reference and monitored clock information is as follows:

| | Reference clock | Monitored Clock | Period Monitor Source |
|---|---|---|---|
| CMU0 | SIRC | RTC_CLK<br>See SMISC for details of the clock source. | SIRC32_1K |
| CMU1 | SIRC | FOSC | SIRC32_1K |
| CMU2 | SIRC | FIRC | SIRC32_1K |
| CMU3 | FIRC | SIRC | SIRC32_1K |
| CMU4 | SIRC | SLOW CLOCK | SIRC32_1K |

***Note:***

- The SIRC32_1K is a common period monitor source for all CMUs in the FC7240 series, so please ensure the SIRC32K is valid to use the Period Monitor mode.
- Not all CMUs can work under STANDBY mode, please see power-related information for details.
- To use the Period Monitor mode and if there is low power entry/exit, CTRL[STOP_EN] must be set (set CTRL[LP_EN] if needed); otherwise, the CMU will gate low power entry.
- CMU4 does not support low power mode because the slow clock is gated under low power mode.
- In the low-power mode, to enable CMU, the reference clock and the monitored clock must be active.
- If CMU is not enabled in the low-power mode, it is recommended to disable CMU before low-power entry and re-enable CMU after low-power wakeup, otherwise, there may be unexpected MIS events introduced by clock recovery, stop abort, etc.
- Because 3 reference clock periods + 3 monitored clock periods are required to enable CMU from the bus domain (register configuration) to the function domain (works), users should guarantee that the time between disabling and re-enabling the CMU must be larger than the required period, otherwise, the internal counters may not be cleared to 0 at the same time, which may cause an MIS event.
- The interrupt flags (MIS and LOC) are generated and cleared at a clock domain that is asynchronous with the bus clock. When the two flags are asserted, software needs to confirm that the flags allocated in the CMU_ST[LOC] and CMU_ST[MIS] fields are cleared successfully by "W1C" before exiting the interrupt service (especially for the system whose bus clock is much faster than the reference/monitor clock).

## 37.3 Features

- If a monitored clock frequency is higher than the high-frequency reference, an MIS event occurs.
- If a monitored clock frequency is lower than the low-frequency reference, an MIS event occurs.
- If a monitored clock does not toggle in the duration of 2*REF_WINDOW*Period of Reference Clock, a LOC event occurs.
- Programmable duration of reference clock cycles.
- Masking of LOC and MIS interrupts.
- Counter clearing by software.

  ***Note:*** Only the internal counters will be cleared. If a LOC event happens, do not use this function because the CMU behaviors after SW clear in this scenario cannot be guaranteed.
- Period Monitor mode supported, monitor programmable times after period start point.
- Can work under Stop/Standby mode.

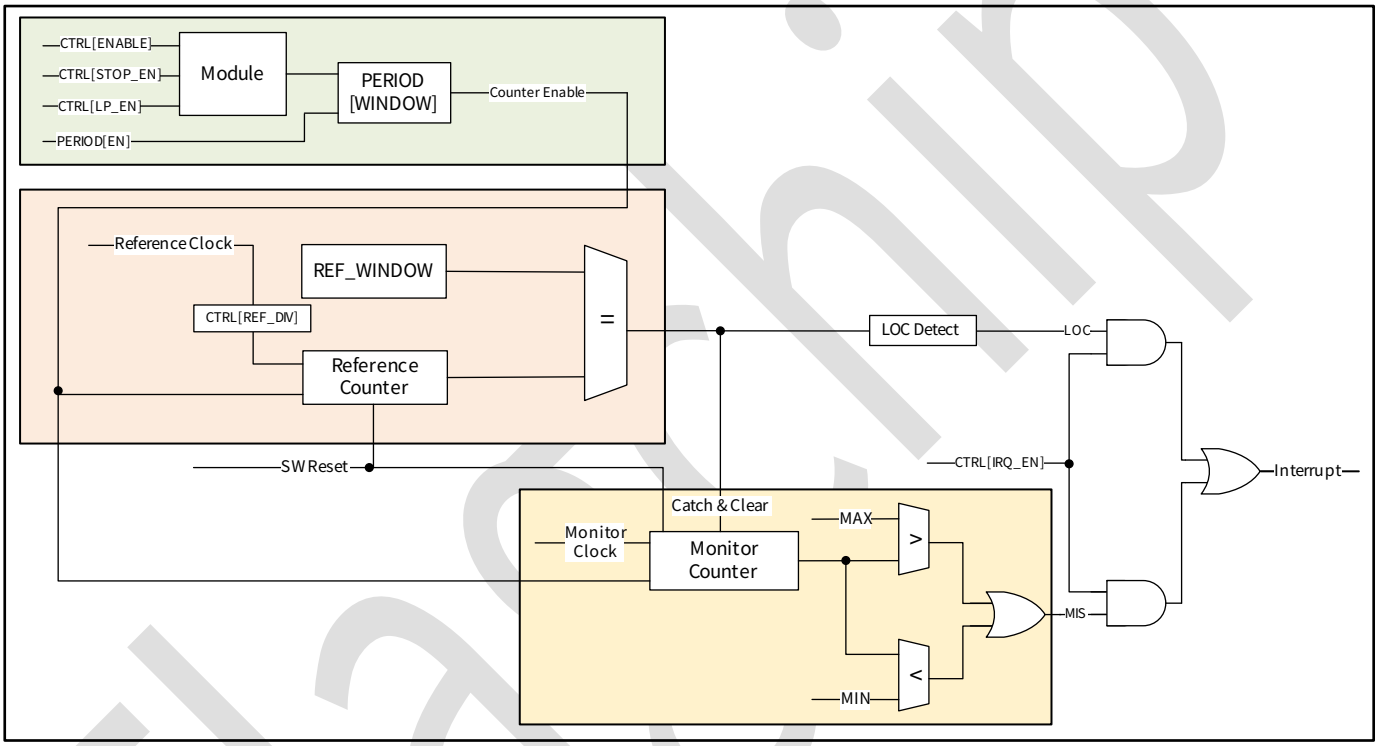
## 37.4 Overview

The CMU is required to ensure clock signal integrity on the chip. The CMU checks if the frequency of a monitored clock is within a programmable frequency range specified by the user. If the frequency is outside the specified limits, it could lead to performance, protocol, and timing failures. The CMU requires a clock that is used as a base reference for the frequency check operation. This reference clock must be within documented limits for correct CMU operation. If a reference clock is outside operating parameters, CMU behavior cannot be guaranteed.

## 37.5 Block Diagram

The following is the block diagram of the CMU module.

**Figure 50. CMU block diagram**




## 37.6 Functional Description

The CMU works after CTRL[ENABLE] is asserted. To work as expected, the CMU requires a fixed number of reference and monitor clock cycles (at least 2 reference+ 2 monitor clock cycles), before hardware operations start, that is, users must guarantee the reference and monitor clock is valid when enabling the CMU.

The CMU performs continuous periodic clock checking of the monitored clock. Each checking window involves the following operations:

- Stage 1 - Reference clock counter runs for REF_WINDOW cycles. The monitored clock counter runs in parallel for the same time duration.
- Stage 2 - After stage 1, the module compares whether the final monitored clock count against the programmed thresholds.

If the monitored clock count is greater than the high threshold value in MAX or lower than the low threshold value in MIN, ST[MIS] will be asserted. If CTRL[IRQ_EN] is set, an interrupt will be generated.

If a monitored clock does not toggle for 2 or more REF_WINDOW cycles of the referenced clock, the ST[LOC] will be asserted. If CTRL[IRQ_EN] is set, an interrupt will be generated.

### 37.6.1 Programming REF_WINDOW

Programming REF_WINDOW must be done after considering application requirements of event response time and required accuracy. A higher value of REF_WINDOW results in a longer checking window leading to better accuracy in monitored clock measurement. A lower value of REF_WINDOW results in a shorter checking window and faster response.

Minimum REF_WINDOW = rounding up value of 6 + 5 × (f (Reference clock) / f (Monitored clock) / 2^REF_DIV).

Maximum REF_WINDOW = MIN value (rounding down) of (0xffffff x (f (Reference clock) / f (monitor clock) / 2^REF_DIV), and 0xffffff.

Example to determine REF_WINDOW:

- f (Reference clock) = 12 MHz
- f (Monitored clock) = 18 MHz
- assume the configuration of REF_DIV is 2

**Min Value Calculation:**

Formula #1: 6 + 5 × (f (Reference clock) / f (Monitored clock) / 2^REF_DIV).

- 6 + 5 × (12 MHz / 18 MHz / 2^2) = 6.83

Therefore, minimum REF_WINDOW = 7. With the configuration, CMU would get the shortest event response time and worst measurement accuracy.

**Max Value Calculation:**

Formula #2: (0xffffff x (f (Reference clock) / f (monitor clock) / 2^REF_DIV), and 0xffffff

- 0xffffff x (12 MHz / 18MHz) / 2^2 = 2796202.5

Limited Value: 0xffffff

MIN (0xffffff, 2796202.5) = 2796202.5

Therefore, maximum REF_WINDOW = 2796202 (0x2aaaaa in hexadecimal). With the configuration, CMU would get the longest event response time and best measurement accuracy.

***Note:***

- The Min and Max values are limited values of REF_WINDOW, but in real situations, from the perspective of measurement accuracy and event response time, it is recommended to use a typical value between Min and Max, like Min value x 100 and Max value /100, etc.
- It takes at least 9 monitor clock cycles for CMU before detecting a real loss behavior.

### 37.6.2 Programming MIN and MAX

MIN/MAX programming must be done considering the following:

- Tolerable monitored clock frequency variation.
- Maximum reference clock frequency variation.
- Inherent module inaccuracy.

The CMU has an expected maximum deviation of +/- CELLING (3, f (Monitored clock) / f (Reference clock)) ideal monitored clock cycles. Example of programming MAX and MIN:

- f (Reference clock) = 12 MHz

- f (Monitored clock) = 18 MHz
- REF_DIV = 0
- REF_WINDOW = 359 (value between Min and Max)
- Ideal Monitored Clock Count = f (Monitored clock) / f (Reference clock) × (REF_WINDOW + 1) × 2^REF_DIV = 540
- Now, consider the combined spec variation of the reference clock and monitored clock = +/- 5%. The inherent inaccuracy of CMU is CELLING (3, f (Monitored clock) / f (Reference clock) cycles.
- Minimum Monitored Clock Count = (Ideal Monitored Clock Count × 0.95) - 3 = 510 (MIN)
- Maximum Monitored Clock Count = (Ideal Monitored Clock Count × 1.05) + 3 = 570 (MAX)

### 37.6.3 Special Mode of CMU

**Period Monitor Mode**

This mode is valid when PERIOD[EN] is set, it is recommended that the PERIOD[WINDOW] > 2 to ensure the LOC function work. If other configurations are reasonable and the CMU is enabled, the CMU will start monitoring at the period start point (positive edge of period monitor source, for period monitor source connection, see Chip-specific Information for details). When the CMU reference counter reaches REF_WINDOW for PERIOD[WINDOW] times, the CMU will stop and wait for the next period start point.

Note that if another period start point occurs during monitoring (period monitor counter < PERIOD[WINDOW]), the period monitor counter will be reset and count from 0.

**Low Power Mode**

The CMU supports work under STOP (CTRL[STOP_EN] needs to be set) or deeper (CTRL[STOP_EN] and CTRL[LP_EN] needs to be set) low power mode (STANDBY mode).

In this mode, the reference clock and monitored clock must be active; otherwise, the CMU would not work (the reference clock does not toggle) or a LOC happens (the monitored clock does not toggle). The CMU would acknowledge the low power entry request immediately.

If these bits are not set before low power entry, the CMU would acknowledge low power requests after all counters are cleared except for the CMU in the Period Monitor mode. To prevent unexpected MIS events after low power wakeup (reference or monitor clock may need sometimes to be stable after low power), the user can set CTRL[RESTART_EN]. If CTRL[RESTART_EN] is set in this situation, the SC[ENABLE] will be automatically cleared by hardware when there is a low power entry request and needs to be re-enabled again after low power wakeup.

### 37.6.4 CMU Programming Sequence

The recommended programming sequence of the CMU is:

1) In any order, do the following:
   - Program REF_WINDOW to define the frequency check window duration in reference clock cycles.
   - Program MIN and MAX to set the permissible frequency range of the monitored clock.
     ***Note:*** REF_WINDOW, MIN, and MAX only support 32-bit access.
   - If needed, program CTRL[REF_DIV] to the prescaler Reference clock.
   - If needed, program CTRL[IRQ_EN] to enable interrupt.
   - If needed, program PERIOD[EN] and PERIOD[WINDOW].
   - If needed, program CTRL[STOP_EN] and CTRL[LP_EN] to work under low power mode.

2) Once the above steps are completed, writing CTRL[ENABLE] = 1 to start the monitor. The ability to write CTRL[REF_DIV], REF_WINDOW, MIN, and MAX are disabled after writing CTRL[ENABLE] = 1. An attempted write to any of those registers is ignored.

3) To reconfigure REF_WINDOW, MIN, and MAX registers, or to stop frequency check operation, write CTRL[ENABLE] = 0.

## 37.7 Register Definition

CMU0 base address : 4003_1000h
CMU1 base address : 4003_2000h
CMU2 base address : 4003_3000h
CMU3 base address : 4003_4000h
CMU4 base address : 4003_5000h

**Register Overview - CMU**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **8h** | CMU_CTRL (Control Register) | 32 | RW | 0000_0000h | page |
| **10h** | CMU_MIN (Minimum Threshold Register) | 32 | RW | 0000_0000h | page |
| **14h** | CMU_MAX (Maximum Threshold Register) | 32 | RW | 00FF_FFFFh | page |
| **18h** | CMU_REF_WINDOW (Reference Window Register) | 32 | RW | 0000_0001h | page |
| **1Ch** | CMU_MON_CNT (Monitor Counter Register) | 32 | RO | 0000_0000h | page |
| **20h** | CMU_ST (Status Register) | 32 | W1C | 0000_0000h | page |
| **24h** | CMU_PERIOD (Period Monitor Mode Configuration Register) | 32 | RW | 0002_0000h | page |

### 37.7.1 CMU_CTRL (Control Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | REF_DIV | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | IRQ_EN | REST ART_ EN | LP_EN | STOP _EN | 0 | SW_RST | ENABLE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMU_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **18-16**<br>**REF_DIV** | RW | Divider for Reference Clock<br>000b - No divider<br>001b - Divided by 2<br>…<br>111b - Divided by 128 |
| **6**<br>**IRQ_EN** | RW | Interrupt Enable<br>If this bit is asserted, either LOC or MIS asserted can lead to an interrupt.<br>1b - Enable<br>0b - Disable |
| **5**<br>**RESTART_EN** | RW | Indicates whether the CMU needs to be re-enabled after exiting from the low-power mode.<br>1b - If STOP_EN and LP_EN are not set, the CMU needs to be re-enabled by setting ENABLE.<br>0b - CMU will keep its status unchanged whether or not STOP_EN and LP_EN are set. |
| **4**<br>**LP_EN** | RW | Standby Mode Enable<br>Requires STOP_EN to be asserted too. |
| **3**<br>**STOP_EN** | RW | Stop Mode Enable |
| **1**<br>**SW_RST** | RW | Software Reset<br>To clear all counters, write 1. This bit will become 0 automatically when the reset is done. |
| **0**<br>**ENABLE** | RW | CMU Enable<br>This bit enables CMU logic.<br>NOTE: Except CTRL0 byte0, all other configured registers can only be written when this bit is 0. |

### 37.7.2 CMU_MIN (Minimum Threshold Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MIN | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MIN | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMU_MIN field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-0 MIN** | RW | The minimum threshold for the monitor counter. |

### 37.7.3 CMU_MAX (Maximum Threshold Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MAX | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAX | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**CMU_MAX field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-0 MAX** | RW | The maximum threshold for the monitor counter. |

### 37.7.4 CMU_REF_WINDOW (Reference Window Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | REF_WINDOW | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REF_WINDOW | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**CMU_REF_WINDOW field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-0 REF_WINDOW** | RW | The sample window for the reference clock. |

### 37.7.5 CMU_MON_CNT (Monitor Counter Register)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MON_CNT | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MON_CNT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMU_MON_CNT field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-0 MON_CNT** | RO | The monitor counter value; can be read if the flag is asserted. |

### 37.7.6 CMU_ST (Status Register)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LOC | 0 | | | | | | MIS | 0 | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | w1c | | | | | | | w1c | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMU_ST field descriptions**

| Field | Type | Description |
|---|---|---|
| **15**<br>**LOC** | W1C | Monitor clock is lost.<br>If the monitor clock does not toggle in the 2*REF_WINDOW*Period of Reference Clock, this flag will be asserted. |
| **8**<br>**MIS** | W1C | Monitor clock is mismatched.<br>When the reference counter reaches REF_WINDOW, and if the monitor counter is less than MIN or greater than MAX, this flag will be asserted. |

### 37.7.7 CMU_PERIOD (Period Monitor Mode Configuration Register)

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | WINDOW | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMU_PERIOD field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16**<br>**WINDOW** | RW | Period Monitor Mode Window<br>In Period monitor mode, CMU starts at a period start point. When CMU completes once operation, the period monitor counter increases by 1. When the counter reaches WINDOW, CMU will stop and wait for the next period start point. |
| **0**<br>**EN** | RW | Period Monitor Mode Enable<br>If this bit is asserted, CMU will sample for WINDOW times.<br>NOTE: If another period start point occurs during this time, the period monitor counter will be cleared and restarted. |

# Chapter 38 Interface Safety Monitor (ISM)

## 38.1 Revision History

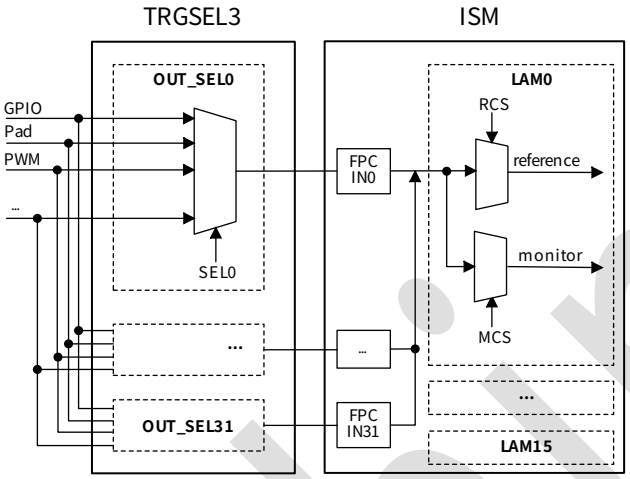
| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 38.2 Chip-specific Information

For Interface Safety Monitor (ISM) input connection, refer to the TRGSEL3 table for details.

**Figure 51. Connection between TRGSEL3 and ISM**




## 38.3 Features

- Support up to 16 channels, each channel has both Monitor and Reference Input. The inputs can come from pad output/feedback input, or from internal IP output.
- Each Monitor/Reference channel supports the glitch mode on rising edges, falling edges or both edges, and counter mode.
- For filter mode, it supports either fixed delay mode (copy the input to output after fixing a preprogrammed delay if the input is not matched with output) or smart delay mode (if the input is not the same as output, decrement (if the counter is not 0) or reset glitch counter, dependent on register setting). The delay option can be independently applied to either edge case. For example, the rising edge uses fixed delay (or no delay), and the falling edge uses smart delay (or no delay). If a rising or falling change while the counter is above 0 but below the threshold, then a rising or falling flag will be set to indicate the glitch that happens during sampling.
- For counter mode, the input is divided down and generates a single pulse when the divider counter reaches the setting. The filter block monitor and reference output polarity can be set independently.
- The monitor block will monitor the reference/monitor output from the filter block.
- The monitor window can be free-running or gated by the filter block reference or monitor output.
- The monitor counter can be clear on the preselected monitor input or reference input edge. The preselected monitor input edge or XOR of monitor input and reference input edge can be used to trigger the event compare during the valid monitor event window. An event is generated if the sample value is above or below the threshold.
- Each Logic Analyzer Module (LAM) channel event output can pass to a 4-bit counter (selected by Event Combiner Module (ECM)). If the event number is above the threshold, it will generate a reset or an interrupt, and can also generate a trigger output.

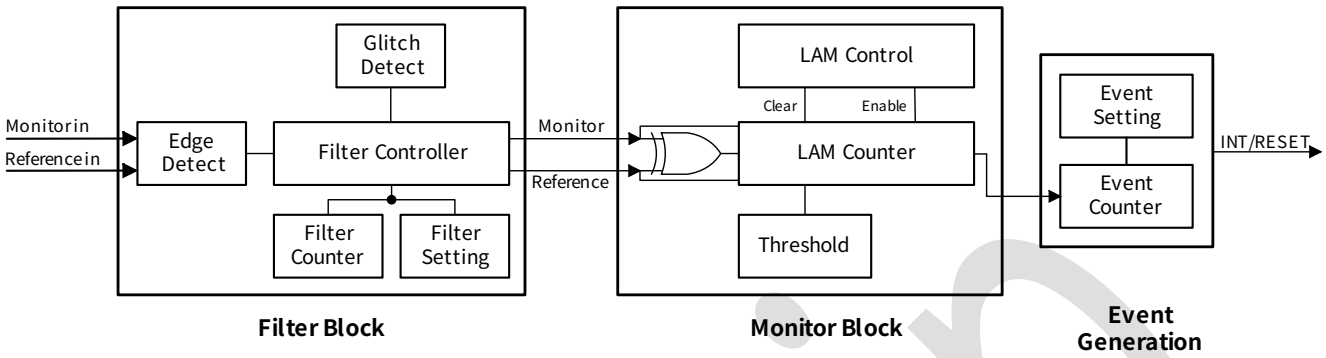
## 38.4 Overview

For the Automotive application, the IO/interface safety is not easy to be handled by software. To improve function safety, the ISM module is added. It can monitor the critical interface/pads output signals (such as critical PWM driver output, and critical communication IO output).

## 38.5 Block Diagram

The following is the block diagram of the ISM module.

**Figure 52. ISM block diagram**

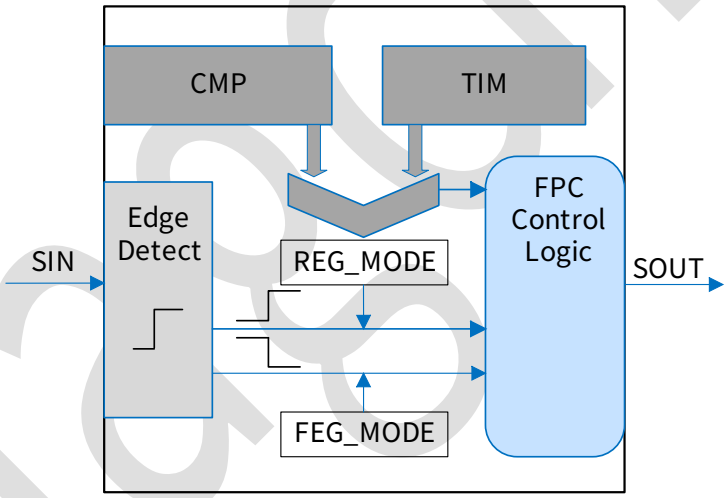



## 38.6 Functional Description

### 38.6.1 Filter & Prescaler Channel (FPC)

16 Filter & Prescaler channels; each connected to a specific pad (PIN), the output of which serves as a monitor or reference signal.

**Figure 53. FPC logic**




The edge mode supports rising/falling edge control separately. For each edge, the below modes are supported. In immediate mode or delay mode, the glitch flag is asserted when a glitch is detected, and this flag can be cleared by SW.

**MODE**

- 00: No filter
- 01: Immediate with filter
- 10: Delay mode.
  - 00/01: Fixed Delay
  - 10: Smart delay with counter decrement when a glitch is detected.
  - 11: Smart delay with counter reset when a glitch is detected.
- 11: Prescaler mode. (When set as prescaler mode, the other edge mode should be set to no filter.)

***Note:*** Except in the Mode "00: No Filter", the value of the FPC_CONFIGn[CMP] bit cannot be set to 0 in the rest three modes, otherwise, the filter threshold will be: 0xffff +1.

**Figure 54. No filter**

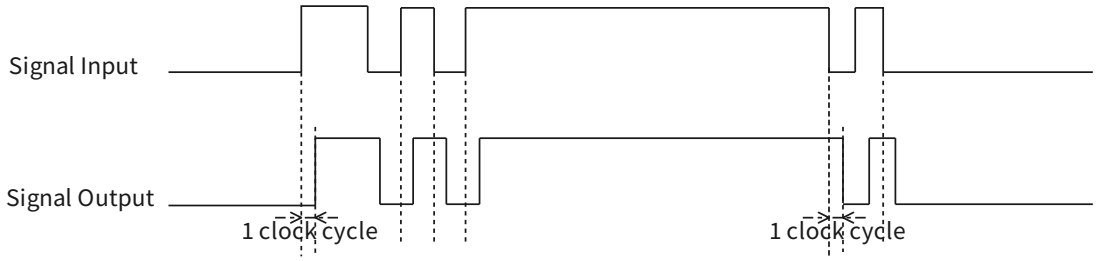



**Figure 55. Immediate mode with filter**

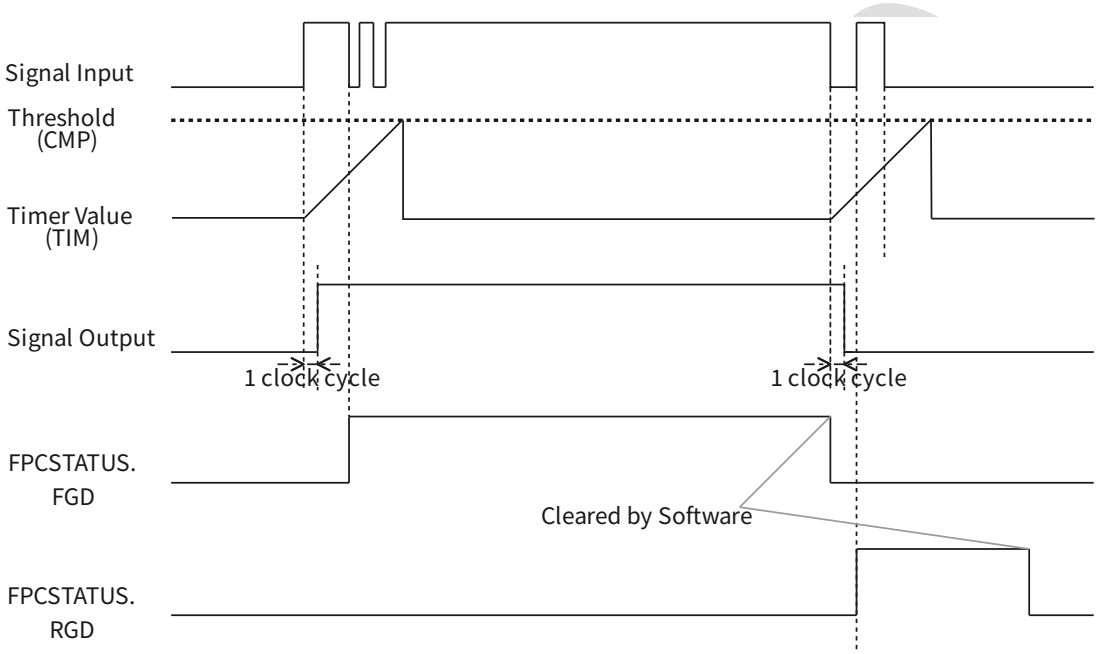



**Figure 56. Smart delay mode with timer decrement**

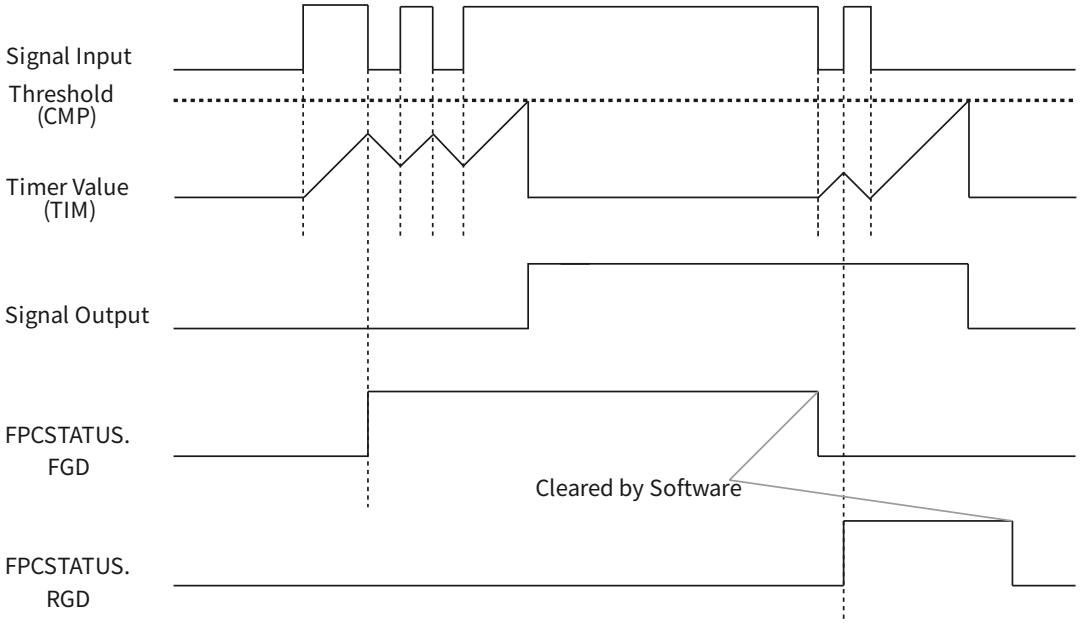


**Figure 57. Smart delay mode with timer reset**

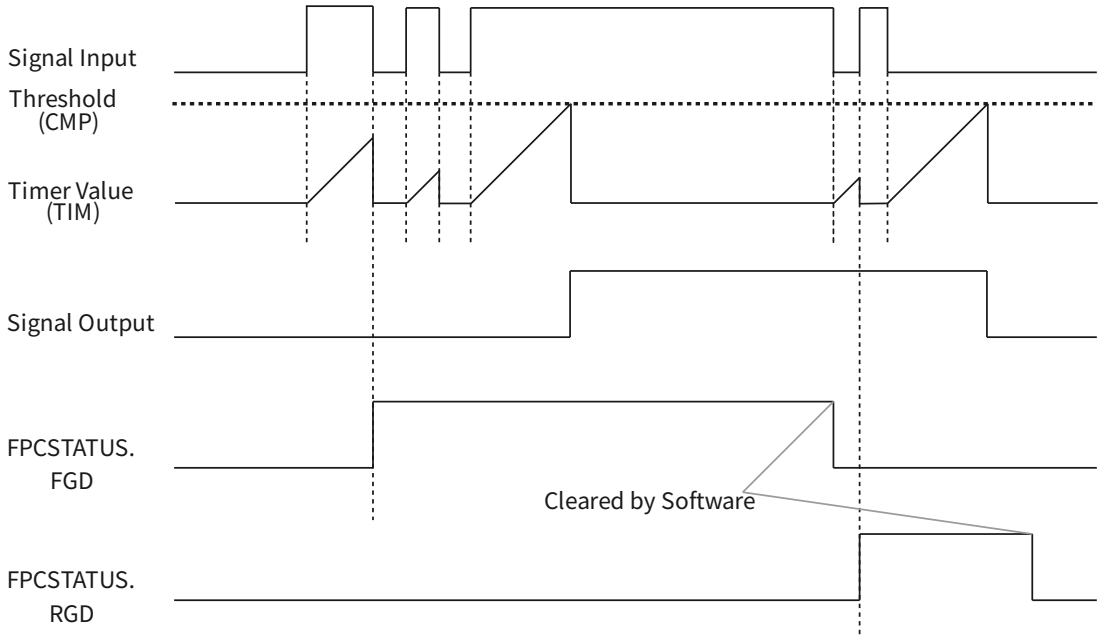
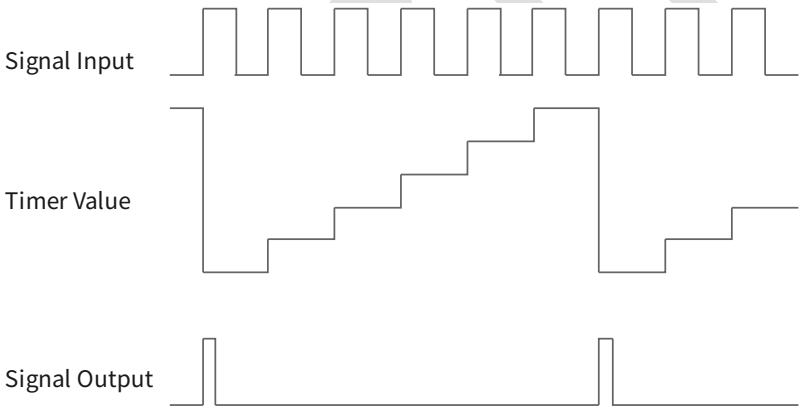



**Figure 58. Prescaler mode**

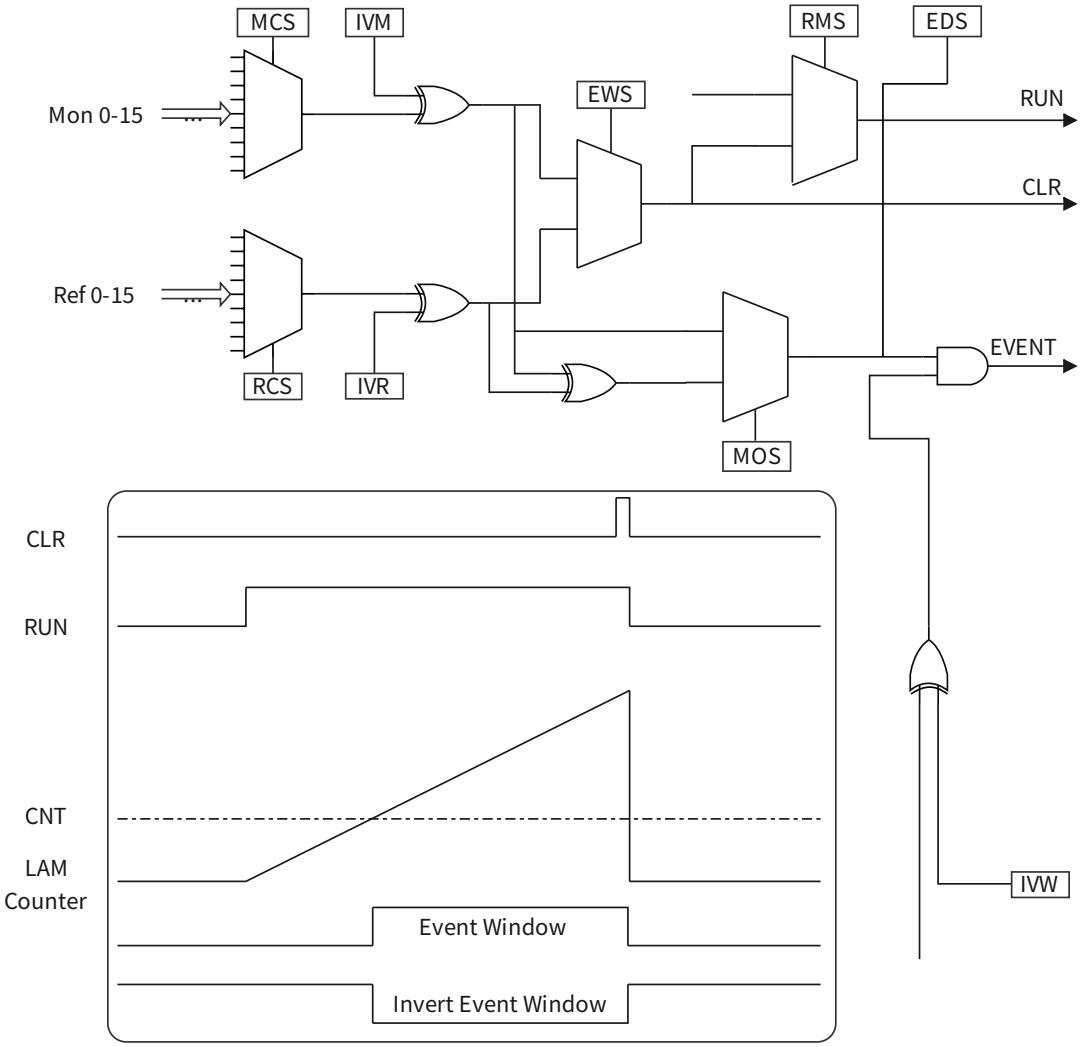



### 38.6.2 Logic Analyzer Module (LAM)

16 Logic Analyzer Modules (LAMs), with each channel comprising a multiplexer to independently select monitor and reference signal inputs, can undertake the compare function. A local event is generated when timing relationships between the selected monitor and reference signals are violated. The LAM is configurable via register settings.

***Note:*** The LAM has a resolution of 1 clock cycle, and it does not support monitoring the duration of a signal or the delay between signals which is equal to 1 clock cycle (after synchronization). The signal or delay whose length is 1 clock cycle (with the event window not inverted) will trigger an event, and the value of the COUNT field of the LAM_STATUS register is 0.

**Figure 59. LAM Logic**




***Note:*** The ISM_LAM_CTRLn[EN] only controls the "LAM Counter" and the counter will be reset to 0 if the EN bit is de-asserted. The event is generated by "Event Window" (generated according to "CNT" and "LAM Counter"), so if the LAM is not expected to work, it's better to de-initialize all control registers for the LAM, especially the ISM_LAM_CONFIGn[IVW] and ISM_LAM_COUNTERn[CNT]. A non-zero value in CNT with IVW asserted in the situation (when EN is de-asserted, or when in stop mode, the LAM counter is stopped by hardware) will enable the "Event Window" always.

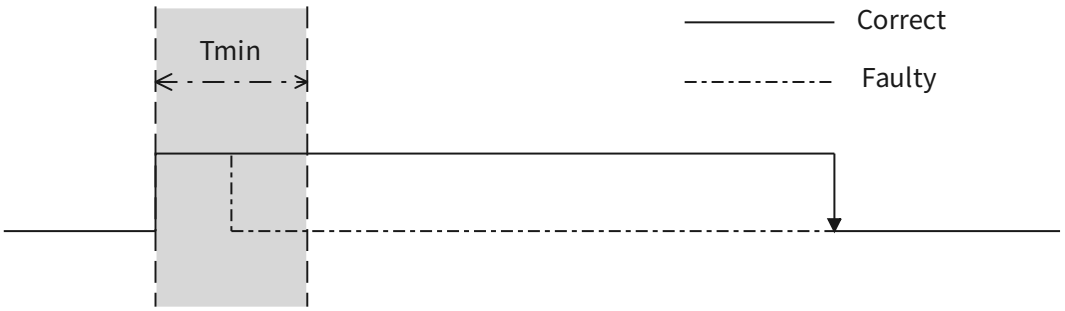
**Example of LAM**

For each channel, the following measurements shall be possible:

- Duty cycle measurement
- Set-up and/or hold time measurement
- Delay window (min, max) with respect to rising or falling edge

The following waveforms provide more examples of typical signal monitoring functionality that could be deployed using the ISM. Some typical configuration settings are also provided in some of the examples.

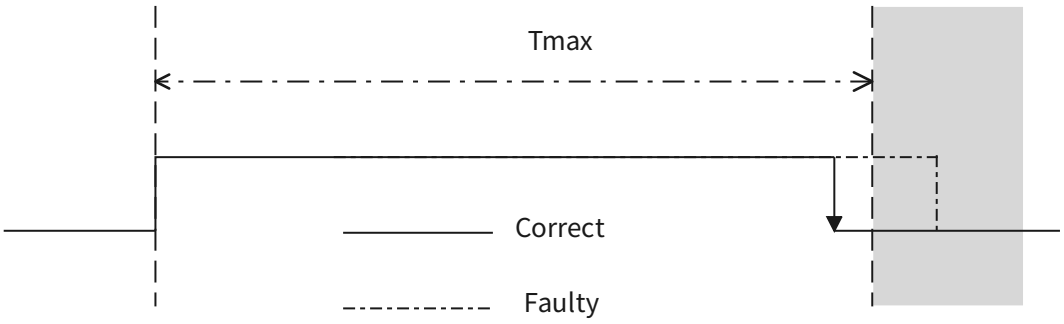
**Example 1 - Pulse or duty cycle too short**




Example settings for LAM block registers:

- LAM_CONFIG.IVR: 0x0; don't care
- LAM_CONFIG.IVM: 0x0; don't invert monitor signal (or invert for non-duty part of the period)
- LAM_CONFIG.MOS: 0x0; monitor signal used for event generation
- LAM_CONFIG.RMS: 0x0; free-running
- LAM_CONFIG.EWS: 0x1; select monitor signal for the event window
- LAM_CONFIG.EDS: 0x9; CLR on the positive edge, EVT on the negative edge
- LAM_CONFIG.IVW: 0x1; invert window, capture events when the counter is below the threshold
- LAM_CONFIG.MCS/RCS; select according to device configuration
- LAM_COUNTER.CNT; select the appropriate threshold (maximum pulse width forbidden. If the pulse width is shorter than or equal to this value, an event will be triggered)

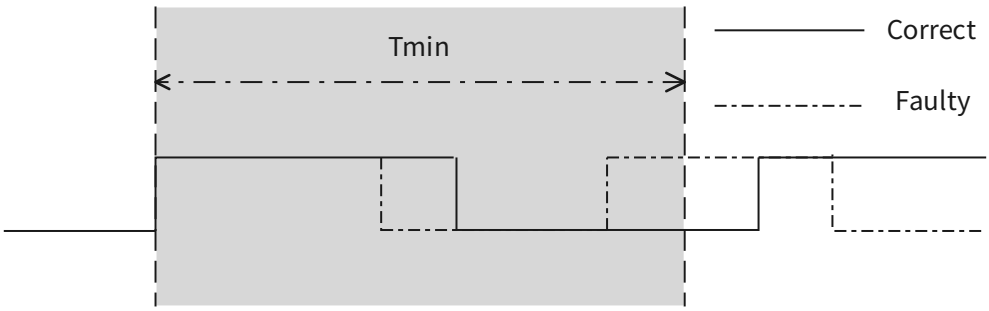
**Example 2 - Pulse or duty cycle too long**




Example settings for LAM block registers:

- LAM_CONFIG.IVR: 0x0; don't care
- LAM_CONFIG.IVM: 0x0; don't invert monitor signal (or invert for non-duty part of period)
- LAM_CONFIG.MOS: 0x0; monitor signal used for event generation
- LAM_CONFIG.RMS: 0x0; free-running
- LAM_CONFIG.EWS: 0x1; select monitor signal for the event window
- LAM_CONFIG.EDS: 0x9; CLR on the positive edge, EVT on the negative edge
- LAM_CONFIG.IVW: 0x0; do not invert window
- LAM_CONFIG.MCS/RCS; select according to device configuration
- LAM_COUNTER.CNT; select the appropriate threshold (maximum pulse width accepted. If the pulse width is longer than this value, an event will be triggered)

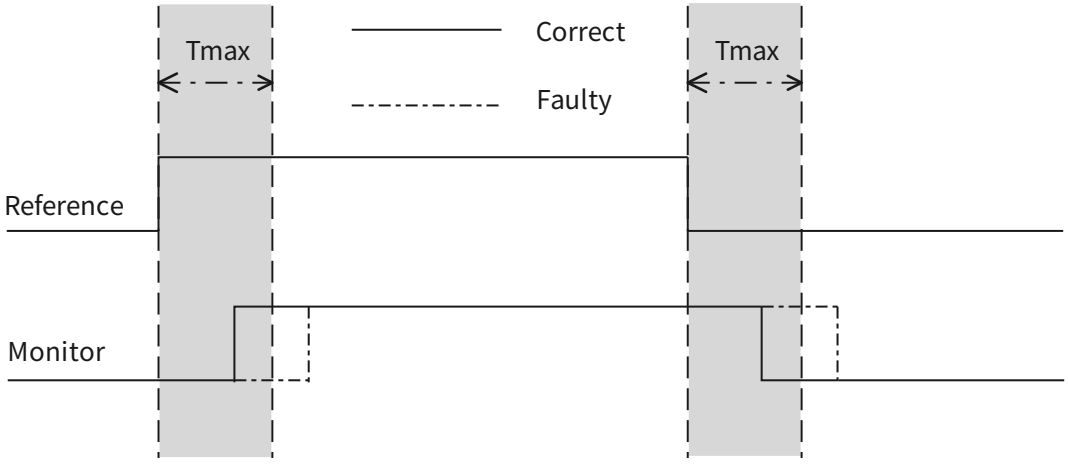
**Example 3 - Period too short**




Example settings for LAM block registers:

- LAM_CONFIG.IVR: 0x0; don't care
- LAM_CONFIG.IVM: 0x0; don't invert monitor signal
- LAM_CONFIG.MOS: 0x0; monitor signal used for event generation
- LAM_CONFIG.RMS: 0x0; free-running
- LAM_CONFIG.EWS: 0x1; select monitor signal for the event window
- LAM_CONFIG.EDS: 0x5; CLR on the positive edge, EVT on the positive edge
- LAM_CONFIG.IVW: 0x1; invert window, capture events when the counter is below the threshold
- LAM_CONFIG.MCS/RCS; select according to device configuration
- LAM_COUNTER.CNT; select the appropriate threshold (maximum period length forbidden. If the period is shorter than or equal to this value, an event will be triggered)

**Example 4 - Diagnosis of Command & Feedback**




Example settings for LAM block registers:

- LAM_CONFIG.IVR: 0x0; don't invert reference signal
- LAM_CONFIG.IVM: 0x0; don't invert monitor signal
- LAM_CONFIG.MOS: 0x1; (MON XOR REF) signal used for event generation
- LAM_CONFIG.RMS: 0x0; free-running
- LAM_CONFIG.EWS: 0x0; select reference signal for the event window
- LAM_CONFIG.EDS: 0xB; CLR on both edges of REF, EVT on the falling edge (of xor'd signal)
- LAM_CONFIG.IVW: 0x0; do not invert window, capture events when the counter is equal to or above the threshold LAM_CONFIG.MCS/RCS; select according to device configuration

- LAM_COUNTER.CNT; set to the max. delay allowed (if the delay between corresponding edges of reference and monitor signals is longer than this value, the event will be triggered)

### 38.6.3 Event Combiner Module (ECM)

The Event Combiner Module (ECM) takes the 16 local event signals (1 from each of the LAM modules) and generates a system event signal from a single, combination, or multiple local event(s). The ECM is configurable via register settings.

## 38.7 Register Definition

ISM base address : 4001_B000h

**Register Overview - ISM**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | ISM_PARAM (Parameter Register) | 32 | RO | 1010_0004h | page |
| **4h** | ISM_CTRL (Control Register) | 32 | RW | 0000_0000h | page |
| **8h** | ISM_E_STATUS (Event Status Register) | 32 | W1C | 0000_0000h | page |
| **Ch** | ISM_E_CTRL (Event Control Register) | 32 | RW | 0000_0000h | page |
| **10h** | ISM_EC_CTRL (Event Control Mode Control Register) | 32 | RW | 0000_0000h | page |
| **100h+n*10h** | ISM_FPC_STATUSn (FPC Status Register) | 32 | W1C | 0000_0000h | page |
| **104h+n*10h** | ISM_FPC_CTRLn (FPC Control Register) | 32 | RW | 0000_0000h | page |
| **108h+n*10h** | ISM_FPC_CONFIGn (FPC Configuration Register) | 32 | RW | 0000_0000h | page |
| **10Ch+n*10h** | ISM_FPC_TIMERn (FPC Timer Register) | 32 | W1C | 0000_0000h | page |
| **400h+n*10h** | ISM_LAM_STATUSn (LAM Status Register) | 32 | W1C | 0000_0000h | page |
| **404h+n*10h** | ISM_LAM_CTRLn (LAM Control Register) | 32 | RW | 0000_0000h | page |
| **408h+n*10h** | ISM_LAM_CONFIGn (LAM Configuration Register) | 32 | RW | 0000_0000h | page |
| **40Ch+n*10h** | ISM_LAM_COUNTERn (LAM Counter Register) | 32 | RW | 0000_0000h | page |

### 38.7.1 ISM_PARAM (Parameter Register)

Indicates configuration for SoC.

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | LAM | | | | | 0 | | FPC | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | ECMC | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**ISM_PARAM field descriptions**

| Field | Type | Description |
|---|---|---|
| **28-24**<br>**LAM** | RO | LAM (Logic Analyzer Module) Number |
| **21-16**<br>**FPC** | RO | FPC (Filter & Prescaler Channel) Number |
| **3-0**<br>**ECMC** | RO | ECM (Event Combination Module) Channel Number |

### 38.7.2 ISM_CTRL (Control Register)

Offset : 4h

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

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | IEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ISM_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**EN** | RW | ISM Enable<br>FPC, LAM and ECM can only be active when ISM enable is asserted.<br>1b - Enable ISM work<br>0b - Disable ISM function |

| | | |
|---|---|---|
| **0**<br>**IEN** | RW | ISM Interrupt Enable<br>When asserted, an ISM interrupt can be output to SoC.<br>1b - Enable ISM to generate a system interrupt<br>0b - ISM cannot generate a system interrupt |

### 38.7.3 ISM_E_STATUS (Event Status Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | ECS | | | |
| W | | | | | | | | | | | | | w1c | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ES | | | | | | | | | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ISM_E_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16**<br>**ECS** | W1C | Event Channel Status<br>Bit n would assert when the ECM channel n event counter reaches the 4-bit threshold (THRL). Write 1 to clear.<br>Bit(n) = 1b - Event happened on ECM channel n<br>Bit(n) = 0b - Event not happened on ECM channel n |
| **15-0**<br>**ES** | W1C | Event Status<br>Bit n would assert when a LAM channel n event happens. Write 1 to clear.<br>Bit(n) = 1b - Event happened on LAM channel n<br>Bit(n) = 0b - Event not happened on LAM channel n |

### 38.7.4 ISM_E_CTRL (Event Control Register)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | ECE | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EE | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ISM_E_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16**<br>**ECE** | RW | Event Channel Enable<br>When bit n asserts, the ECM channel n can trigger a system event.<br>Bit(n) = 1b - Enable ECM channel n to generate a system event<br>Bit(n) = 0b - ECM channel n cannot generate a system event |
| **15-0**<br>**EE** | RW | Event Enable<br>When bit n asserts, the LAM channel n can trigger a system event.<br>Bit(n) = 1b - Enable LAM channel n to generate a system event<br>Bit(n) = 0b - LAM channel n cannot generate a system event |

### 38.7.5 ISM_EC_CTRL (Event Control Mode Control Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | THRL_3 | | | | SEL_3 | | | | THRL_2 | | | | SEL_2 | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | THRL_1 | | | | SEL_1 | | | | THRL_0 | | | | SEL_0 | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ISM_EC_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-28**<br>**THRL_3** | RW | Determines the threshold of the ECM channel 3 counter value.<br>0000b - Counter is disabled and the channel cannot output an event.<br>0001b-1111b - Threshold count value. When the number of events reaches the value, the channel event will be triggered. |

| | | |
|---|---|---|
| **27-24**<br>**SEL_3** | RW | Determines which LAM channel event can be routed to counter ECM channel 3. |
| **23-20**<br>**THRL_2** | RW | Determines the threshold of the ECM channel 2 counter value.<br>0000b - Counter is disabled and the channel cannot output an event.<br>0001b-1111b - Threshold count value. When the number of events reaches the value, the channel event will be triggered. |
| **19-16**<br>**SEL_2** | RW | Determines which LAM channel event can be routed to counter ECM channel 2. |
| **15-12**<br>**THRL_1** | RW | Determines the threshold of the ECM channel 1 counter value.<br>0000b - Counter is disabled and the channel cannot output an event.<br>0001b-1111b - Threshold count value. When the number of events reaches the value, the channel event will be triggered. |
| **11-8**<br>**SEL_1** | RW | Determines which LAM channel event can be routed to counter ECM channel 1. |
| **7-4**<br>**THRL_0** | RW | Determines the threshold of the ECM channel 0 counter value.<br>0000b - Counter is disabled and the channel cannot output an event.<br>0001b-1111b - Threshold count value. When the number of events reaches the value, the channel event will be triggered. |
| **3-0**<br>**SEL_0** | RW | Determines which LAM channel event can be routed to counter ECM channel 0. |

### 38.7.6 ISM_FPC_STATUSn (FPC Status Register)

Offset : 100h+N*10h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | RGD | FGD |
| W | | | | | | | | | | | | | | | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ISM_FPC_STATUSn field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **1**<br>**RGD** | W1C | FPC Channel Rising Glitch Detected. Write 1 to clear. |
| **0**<br>**FGD** | W1C | FPC Channel Falling Glitch Detected. Write 1 to clear. |

### 38.7.7 ISM_FPC_CTRLn (FPC Control Register)

Offset : 104h+N*10h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | IEN | EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ISM_FPC_CTRLn field descriptions**

| Field | Type | Description |
|---|---|---|
| **1**<br>**IEN** | RW | FPC Glitch Detect Interrupt Enable. |
| **0**<br>**EN** | RW | FPC Channel Enable<br>FPC can only be active when this bit and ISM enable are both asserted. |

### 38.7.8 ISM_FPC_CONFIGn (FPC Configuration Register)

Offset : 108h+N*10h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | FEG | | FED | | REG | | RED | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ISM_FPC_CONFIGn field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-22**<br>**FEG** | RW | FPC Falling Edge Detect Mode<br>00b - No filter mode. Copy input to SOUT directly, which is LAM input.<br>01b - Immediate filter mode.<br>10b - Delay mode.<br>11b - Prescaler mode. |
| **21-20**<br>**FED** | RW | FPC Falling Edge Delay Mode<br>00b/01b - Fixed delay mode.<br>10b - Smart delay mode. The counter is decremented when a glitch happens.<br>11b - Smart delay mode. The counter is reset when a glitch happens. |
| **19-18**<br>**REG** | RW | FPC Rising Edge Detect Mode<br>00b - No filter mode. Copy input to SOUT directly, which is LAM input.<br>01b - Immediate filter mode.<br>10b - Delay mode.<br>11b - Prescaler mode. |
| **17-16**<br>**RED** | RW | FPC Rising Edge Delay Mode<br>00b/01b - Fixed delay mode.<br>10b - Smart delay mode. The counter is decremented when a glitch happens.<br>11b - Smart delay mode. The counter is reset when a glitch happens. |
| **15-0**<br>**CMP** | RW | FPC threshold value that is compared with the 16-bit timer. |

### 38.7.9 ISM_FPC_TIMERn (FPC Timer Register)

Offset : 10Ch+N*10h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIM | | | | | | | | | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ISM_FPC_TIMERn field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**TIM** | W1C | FPC Timer Value<br>Show the value of the timer of the filter and prescaler channel. Write to this register will clear timer counter. |

### 38.7.10 ISM_LAM_STATUSn (LAM Status Register)

Offset : 400h+N*10h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OVFL | 0 | | | | | | | COUNT | | | | | | | |
| W | w1c | | | | | | | | w1c | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COUNT | | | | | | | | | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ISM_LAM_STATUSn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**OVFL** | W1C | LAM Timer Overflow Flag<br>Asserts when the LAM counter reaches 2'hFF_FFFF. Write 1 to clear. |
| **23-0**<br>**COUNT** | W1C | LAM Event Window Count Value<br>Gets the value of the LAM counter when a LAM event is triggered. Write any to clear. |

### 38.7.11 ISM_LAM_CTRLn (LAM Control Register)

Offset : 404h+N*10h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | IEN | EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ISM_LAM_CTRLn field descriptions**

| Field | Type | Description |
|---|---|---|
| **1**<br>**IEN** | RW | LAM Channel Overflow Interrupt Enable |
| **0**<br>**EN** | RW | LAM Channel Enable<br>LAM can only be active when this bit and ISM enable are both asserted. |

### 38.7.12 ISM_LAM_CONFIGn (LAM Configuration Register)

Offset : 408h+N*10h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | RCS | | | | 0 | | | | MCS | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | IVW | EDS | | | | 0 | | | EWS | RMS | MOS | IVM | IVR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ISM_LAM_CONFIGn field descriptions**

| Field | Type | Description |
|---|---|---|
| **27-24**<br>**RCS** | RW | Reference input signal selection of LAM from FPC. |
| **19-16**<br>**MCS** | RW | Monitor input signal selection of LAM from FPC. |
| **12**<br>**IVW** | RW | Invert Event Window of LAM Channel<br>0b - Event window non-inverted.<br>1b - Event window inverted. |

| | | |
|---|---|---|
| **11-8**<br>**EDS** | RW | Event Window Active Edge Selection<br>This bit field determines which active edges of the monitor and reference signals are used for the event window generation.<br>0000b - Neither edge used to clear the event window counter. Neither edge used to gate event generation.<br>0001b - Positive edge used to clear the event window counter. Neither edge used to gate event generation.<br>0010b - Negative edge used to clear the event window counter. Neither edge used to gate event generation.<br>0011b - Either edge used to clear the event window counter. Neither edge used to gate event generation.<br>0100b - Neither edge used to clear the event window counter. Positive edge used to gate event generation.<br>0101b - Positive edge used to clear the event window counter. Positive edge used to gate event generation.<br>0110b - Negative edge used to clear the event window counter. Positive edge used to gate event generation.<br>0111b - Either edge used to clear the event window counter. Positive edge used to gate event generation.<br>1000b - Neither edge used to clear the event window counter. Negative edge used to gate event generation.<br>1001b - Positive edge used to clear the event window counter. Negative edge used to gate event generation.<br>1010b - Negative edge used to clear the event window counter. Negative edge used to gate event generation.<br>1011b - Either edge used to clear the event window counter. Negative edge used to gate event generation.<br>1100b - Neither edge used to clear the event window counter. Either edge used to gate event generation.<br>1101b - Positive edge used to clear the event window counter. Either edge used to gate event generation.<br>1110b - Negative edge used to clear the event window counter. Either edge used to gate event generation.<br>1111b - Either edge used to clear the event window counter. Either edge used to gate event generation. |
| **4**<br>**EWS** | RW | Event Window Selection<br>0b - Event window generation is determined from the reference signal.<br>1b - Event window generation is determined from the monitor signal. |
| **3**<br>**RMS** | RW | Run Mode Selection<br>0b - Event window generation is free-running.<br>1b - Event window generation is gated with the monitor or reference signal. |

Flagchip

| | | |
|---|---|---|
| **2**<br>**MOS** | RW | Monitor Source Selection<br>0b - Monitor signal is sourced directly from FPC monitor channel<br>1b - Monitor signal is EXOR'd with FPC reference channel. |
| **1**<br>**IVM** | RW | Invert Monitor<br>0b - Do not invert the monitor signal from FPC.<br>1b - Invert the monitor signal from FPC. |
| **0**<br>**IVR** | RW | Invert Reference<br>0b - Do not invert the reference signal from FPC.<br>1b - Invert the reference signal from FPC. |

### 38.7.13 ISM_LAM_COUNTERn (LAM Counter Register)

Offset : 40Ch+N*10h (N : 0 ~ 15)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CNT | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ISM_LAM_COUNTERn field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-0**<br>**CNT** | RW | Event Window Count Threshold<br>This bit field contains the value of the counter at which the event window becomes active (before optional inversion). |

# Chapter 39 Fault Control and Safety Management Unit (FCSMU)

## 39.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | Changed the access attribute from "RO" to "RW" for the register *FCSMU_CRC_RES* in the register overview table in Section *Register Definition* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 39.2 Chip-specific Information

On FC7240F2M, FCSMU_PIN0/1 are controlled by STCU during LBIST if this feature is enabled in NVR load.

## 39.3 Features

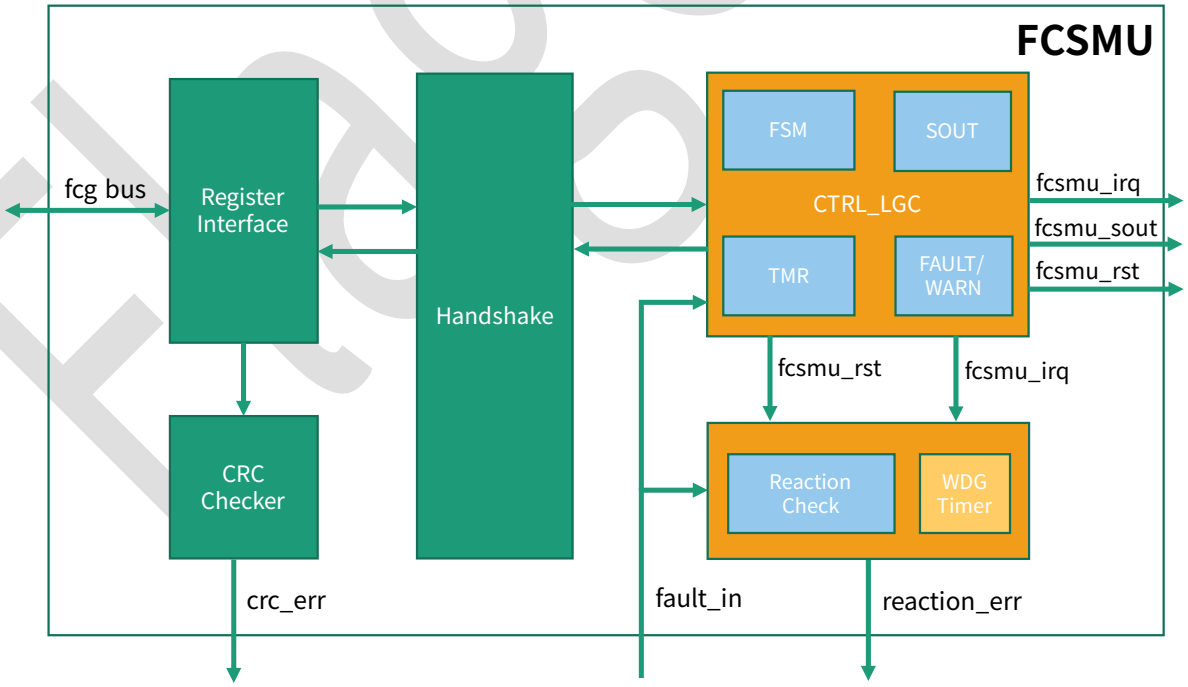
- Support up to 32 fault channels.
- Each channel can be enabled/disabled respectively.
- Each channel can be configured to generate an interrupt, reset, status output (SOUT), or no reaction.
- Status registers to indicate which fault is detected.
- Support FCSMU_PIN0/1 output pins to indicate if the device is in a fault state or not.
- Configurable WARNING state of FCSMU.
- Support configuration lock to avoid unexpected configuration or hack.
- Support configuration watchdog timeout.
- Support self-check feature including configuration register CRC check and reaction check. FCSMU can be reset when a self-check error is detected.
- Support fault injection into fault channels by writing INJECT register.

## 39.4 Overview

The Fault Control and Safety Management Unit (FCSMU) module is used to manage faults in the device and put the device into a safe state via generating an interrupt or a reset when any fault is detected. The FCSMU offers an efficient safety management strategy that can improve function safety of the device.

## 39.5 Block Diagram

**Figure 60. FCSMU block diagram**




## 39.6 Functional Description

The FCSMU is used to monitor the faults in the device and offers an efficient safety management strategy that can improve the function safety of the device. The FCSMU supports many fault reactions when any fault is detected. They are:

- Warn interrupt

- Fault interrupt
- Functional reset
- Indicates FCSMU status via FCSMU_PIN0/1

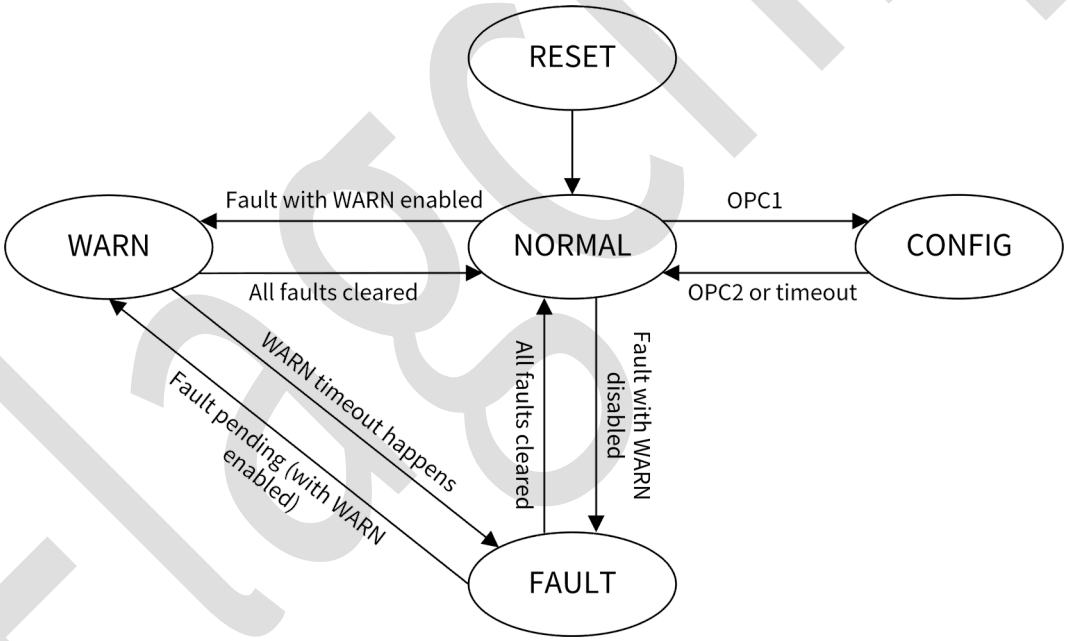
The FCSMU supports a self-check strategy to ensure the correctness of FCSMU behaviors.

- 32-bit CRC check for configuration registers value. This can detect unexpected register value changes. Can generate an interrupt or a reset when a CRC error is detected which is configurable.
- Check the reaction of the FCSMU. If a fault signal is asserted and the channel is enabled with reaction, but the FCSMU outputs no reaction within a fixed time, this will generate a full chip reset request. This feature is enabled by default and cannot be disabled.

### 39.6.1 FCSMU State Switch

The FCSMU is controlled by the Finite State Machine (FSM) as shown in the diagram.

- NORMAL: Indicates there is no fault detected.
- CONFIG: The configuration registers only can be written in this state.
- WARN: A fault with WARN state enabled is detected. FCSMU generates a warn interrupt along with entering WARN state.
- FAULT: A fault with WARN disabled is detected or WARN timeout happens before the fault is cleared.




#### *39.6.1.1 Switch from NORMAL to CONFIG State*

The FCSMU is in NORMAL state after reset. Before writing the configuration registers, it is needed to put the FCSMU into CONFIG state. Otherwise, writing configuration state will cause a hard fault.

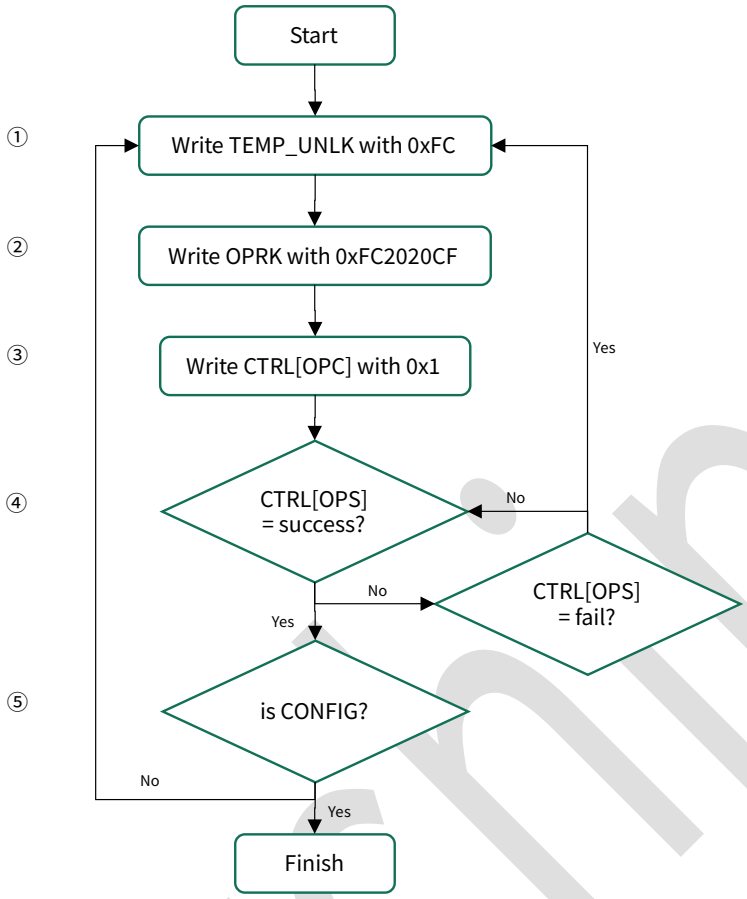
① Unlock CONFIG state by writing TEMP_UNLK register with 0xFC.

② Unlock OPC1 by writing OPRK register with 0xFC2020CF.

③ Writing CTRL[OPC] with 0x1 for OPC1.

④ Check CTRL[OPS] is successful. If not, please go to ①.

⑤ Check FCSMU is in CONFIG state by reading STATUS[STAT]. If not, please go to ①.




### *39.6.1.2 Switch from CONFIG to NORMAL State*

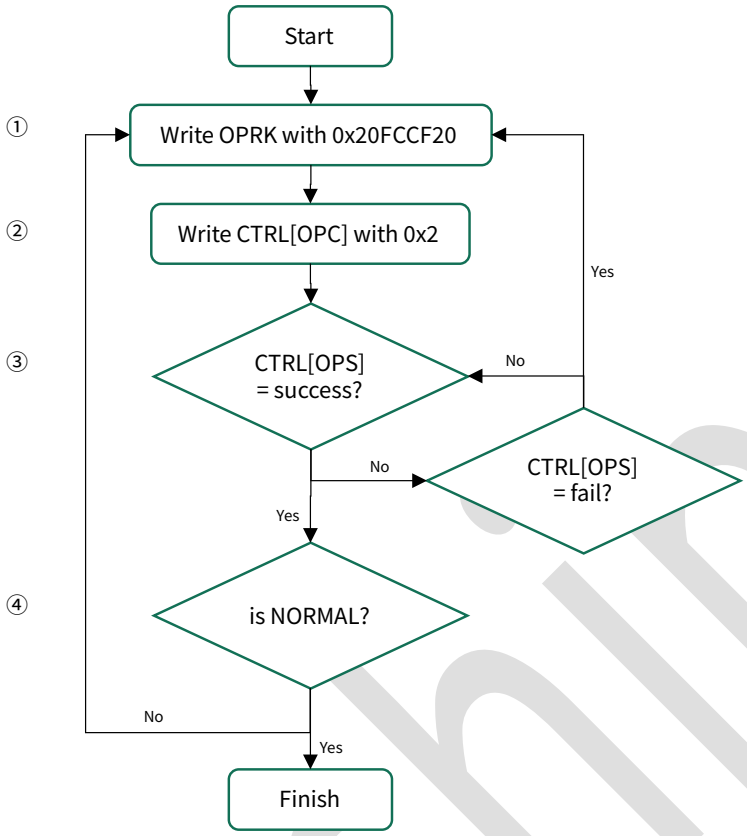
After finishing configuring the FCSMU, you need to switch from CONFIG state to NORMAL state, then the configuration will take effect. The steps are:

① Unlock OPC2 by writing OPRK register with 0x20FCCF20.

② Writing CTRL[OPC] with 0x2 for OPC2.

③ Check CTRL[OPS] is successful. If not, please go to ①.

④ Check FCSMU is in NORMAL state by reading STATUS[STAT]. If not, please go to ①.




### *39.6.1.3 Switch from WARN/FAUTL to NORMAL State*

Each fault channel of the FCSMU supports hardware clear and software clear. For a hardware clear fault, the FCSMU goes to FAULT state when the fault is detected and goes back to the NORMAL state when the fault is cleared automatically without software intervention. For a software clear fault, the FCSMU goes to FAULT state when the fault is detected and goes back to the NORMAL state when the fault is cleared and software clears the fault status register manually. Below are the steps to clear a software-clearable fault：

① Resolve the source of the software clearable fault.

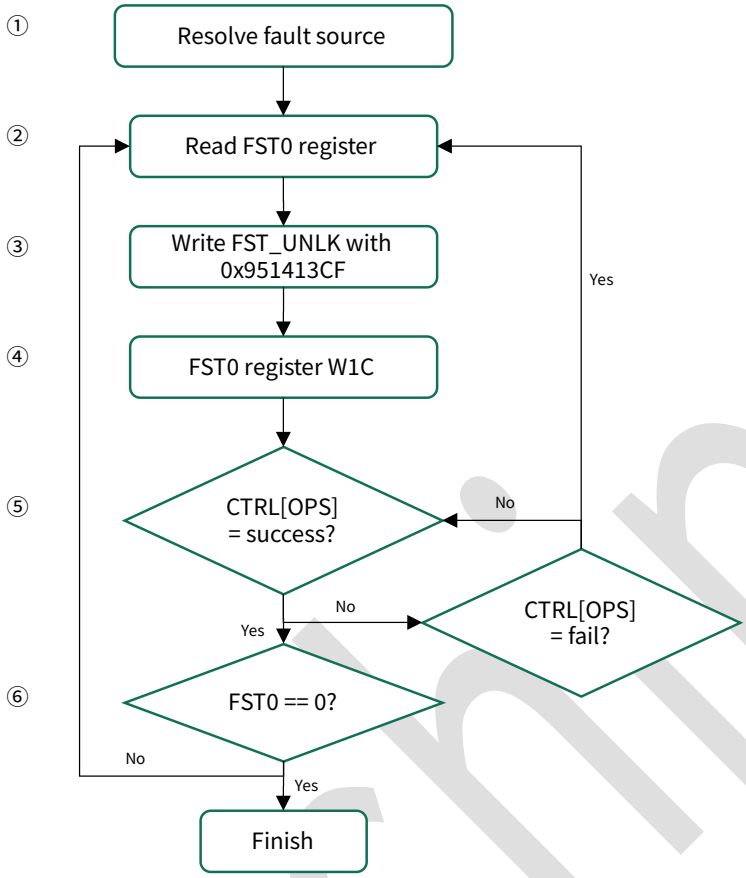
② Get the fault channel by reading FST0 register.

③ Unlock FST0 register by writing FST_UNLK with 0x951413CF.

④ Clear the FST0 register by W1C.

⑤ Check CTRL[OPS] is successful. If not, please go to ②.

⑥ Check all faults are cleared by reading FST0. If not, please go to ②.




### 39.6.2 FCSMU Interrupt and Functional Reset

The FCSMU is in the NORMAL state at reset and all fault channels are disabled by default. Software needs to follow the steps below to configure FCSMU.

① Switch from NORMAL state to CONFIG state.

② Configure the maximum interval for FCSMU in WARN state.

③ Enable WARN interrupt by configuring WARNING_IEN0 registers for specified channels.

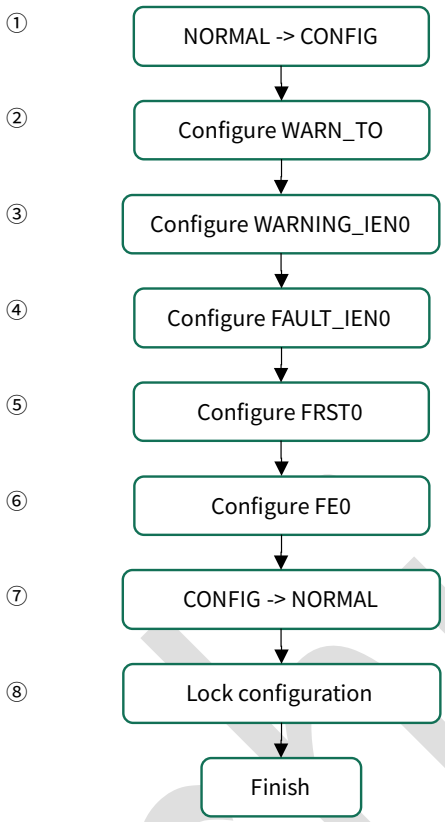
④ Enable FAULT interrupt by configuring FAULT_IEN0 registers for specified channels.

⑤ Enable functional reset by configuring FRST0 registers for specified channels.

⑥ Enable specified channels by configuring FE0 register.

⑦ Switch from CONFIG state to NORMAL state.

⑧ (optional)Lock the configuration by configuring TEMP_UNKL and PERMNT_LOCK.




### 39.6.3 FCSMU Configuration CRC Check

To increase function safety, the FCSMU supports CRC check for configuration registers which can self-check if the configuration is changed unexpectedly. Once the CRC error is detected, the RGM can reset the device including FCSMU (For more information, please refer to the RGM chapter). The below shows two CRC generation flows: one is software CRC generation, and the other is trigger CRC generation.

**Table 64. Configuration CRC check covered registers**

| Index | Register Name | Mask |
|---|---|---|
| 1 | CTRL | 0xE0000200 |
| 2 | SOCTRL | 0xFFFFFFFF |
| 3 | FCCC0 | 0xFFFFFFFF |
| 4 | FRST0 | 0xFFFFFFFF |
| 5 | FE0 | 0xFFFFFFFF |
| 6 | WARNING_EN0 | 0xFFFFFFFF |
| 7 | WARNING_TO | 0xFFFFFFFF |
| 8 | CFG_TO | 0xFFFFFFFF |
| 9 | IRQ_EN | 0xFFFFFFFF |
| 10 | STMR | 0xFFFFFFFF |
| 11 | WARNING_IEN0 | 0xFFFFFFFF |
| 12 | FAULT_IEN0 | 0xFFFFFFFF |
| 13 | SOUT_EN0 | 0xFFFFFFFF |

**1) Software CRC Generation Flow:**

① Finish FCSMU configuration.

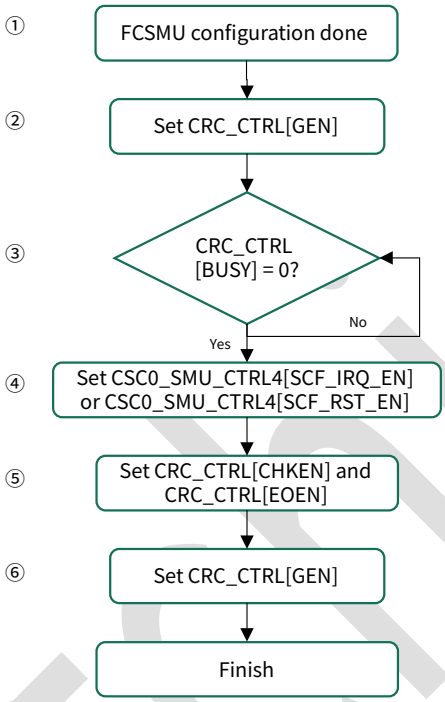
② Software starts a CRC generation by setting CRC_CTRL[GEN]. This step will generate an original CRC result.

③ Wait for CRC generation done by polling CRC_CTRL[BUSY] to be 0.

④ Enable CRC interrupt or reset by setting CSC0_SMU_CTRL4[SCF_IRQ_EN] or CSC0_SMU_CTRL4[SCF_RST_EN].

⑤ Enable CRC error checker by setting CRC_CTRL[CHKEN] and CRC_CTRL[EOEN].

⑥ After ①~⑤ done, each time you want to start a software CRC check, you just need to write CRC_CTRL[GEN] with 1.




**2) Trigger CRC Generation Flow:**

① Finish FCSMU configuration.

② Configure CSC0_AONCLKSR[AONCLKSEL] to select the trig source.

③ Software starts a CRC generation by setting CRC_CTRL[GEN]. This step will generate an original CRC result.

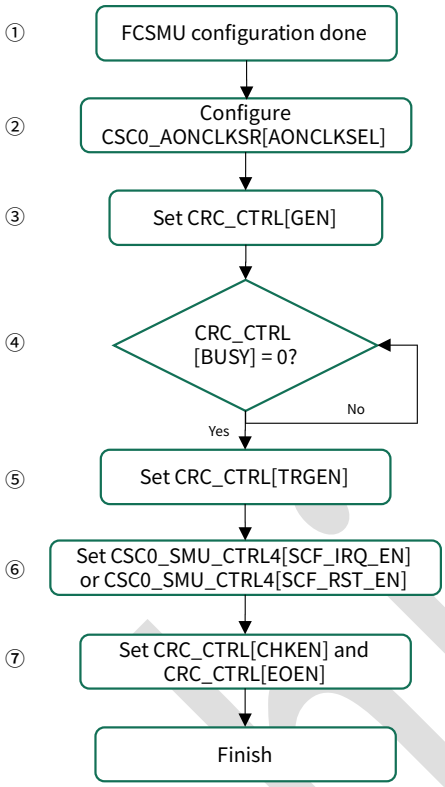
④ Wait for CRC generation done by polling CRC_CTRL[BUSY] to be 0.

⑤ Enable trigger mode by setting CRC_CTRL[TRGEN].

⑥ Enable CRC interrupt or reset by setting CSC0_SMU_CTRL4[SCF_IRQ_EN] or CSC0_SMU_CTRL4[SCF_RST_EN].

⑦ Enable CRC error checker by setting CRC_CTRL[CHKEN] and CRC_CTRL[EOEN].

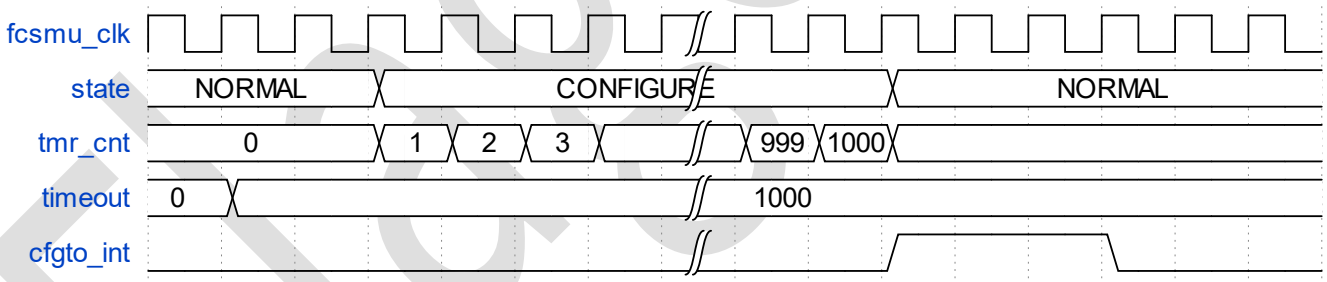
Flagchip




### 39.6.4 FCSMU Configuration Timeout

The FCSMU will go from the configuration state to the normal state automatically and generate an interrupt if software does not finish the configuration in time specified by CFG_TO register. When configuration timeout happens, the configuration just now will not take effect. This is a strategy to avoid unexpected configuration and hack.

**Figure 61. FCSMU configuration timeout**




### 39.6.5 FCSMU Fault Management

#### *39.6.5.1 FCSMU Warning interrupt*

The FCSMU goes from the normal state to the warning state and generates a warning interrupt if the warning state is enabled for the fault channel. The FCSMU goes from warning state to normal state if the fault is cleared before warning timeout. Otherwise, the FCSMU goes from the warning state to the fault state.

**Figure 62. FCSMU goes to fault state after warning timeout**

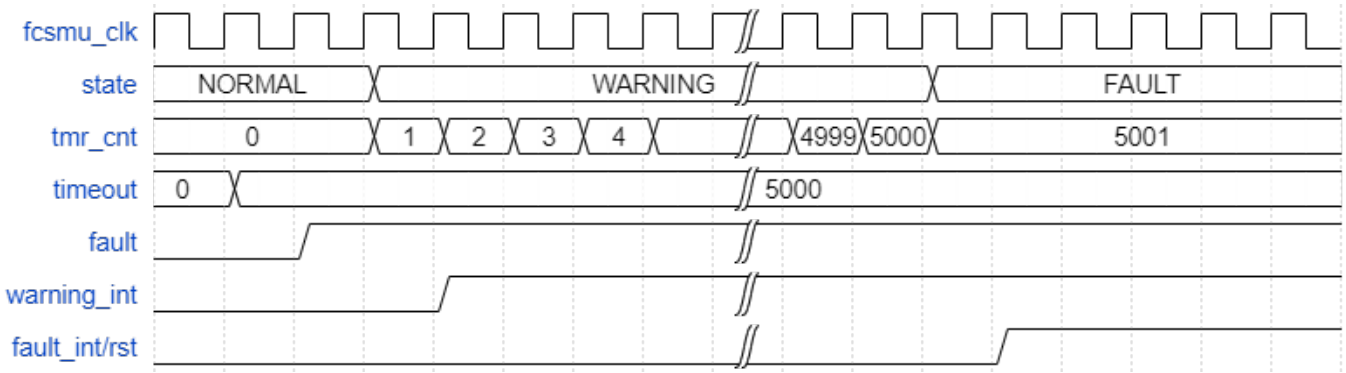
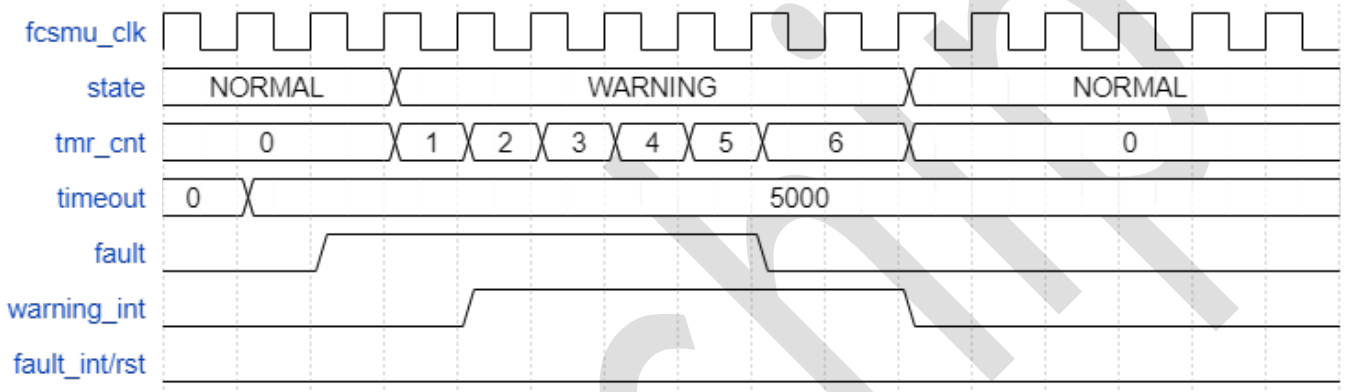


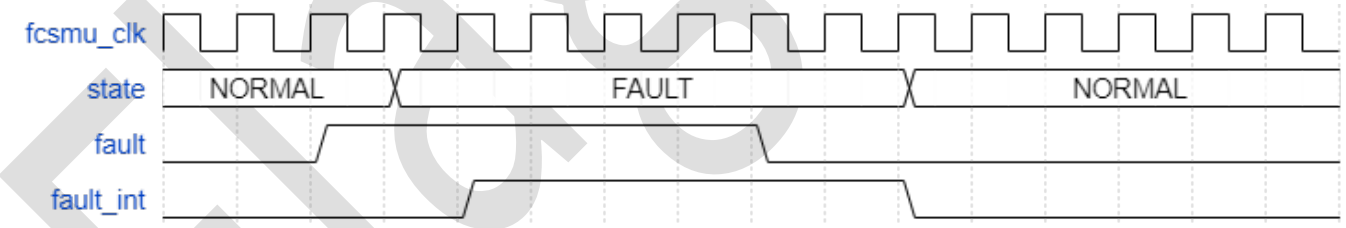

**Figure 63. FCSMU clears fault in warning state**




### *39.6.5.2 FCSMU Fault Interrupt*

The FCSMU goes from the normal state to the fault state and generates a fault interrupt if the warning state is not enabled and the fault channel is configured as interrupt not reset. The FCSMU goes from the fault state to the normal state if the fault is cleared.

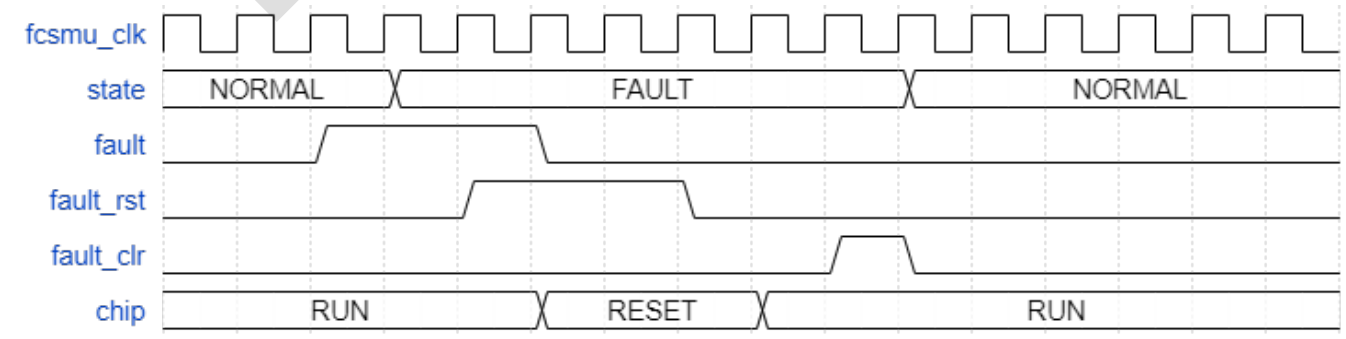
**Figure 64. FCSMU goes to fault state with interrupt output**




### *39.6.5.3 FCSMU Fault Reset*

The FCSMU goes from the normal state to the fault state and generates a fault reset if the warning state is not enabled and the fault channel is configured as reset. The FCSMU goes from the fault state to the normal state if the fault is cleared.

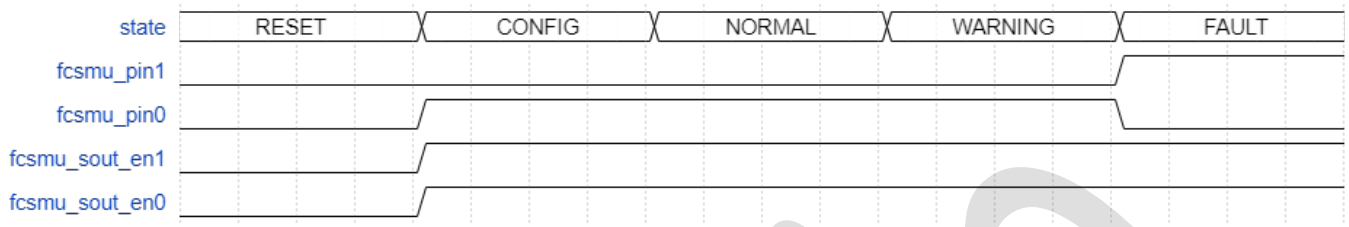
**Figure 65. FCSMU goes to fault state with reset output**



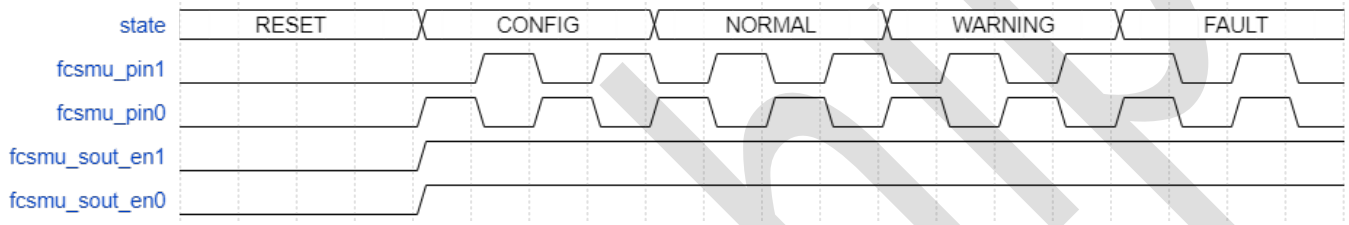
#### *39.6.5.4 FCSMU SOUT*

The FCSMU supports FCSMU_PIN0/1 output pins to externally indicate if the device is in the fault state or not in redundant protocols. They are BI-STABLE, DUAL-RAIL, TIME-SWITCH, FAULT-TOGGLE, and TIME-DUAL-RAIL. The FCSMU_PIN0/1 are encoded as shown below (Note that the timing diagrams do not imply transitions between one condition and another).

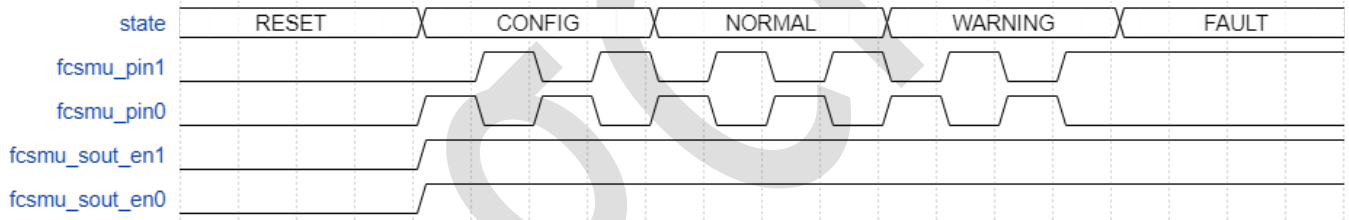
1) BI-STABLE(SOCTRL[POLSW] is 0)



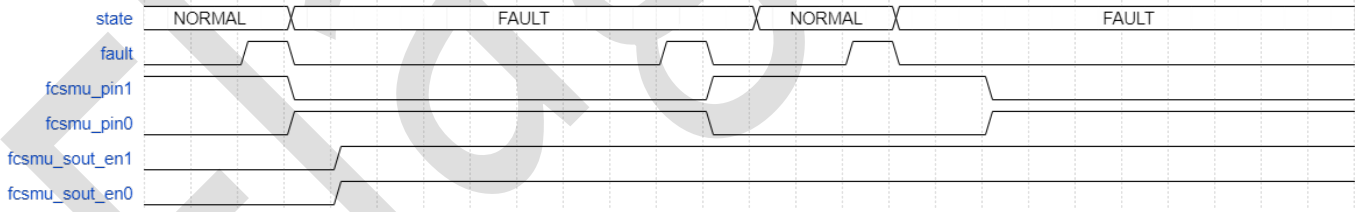

2) DUAL-RAIL(SOCTRL[POLSW] is 0)



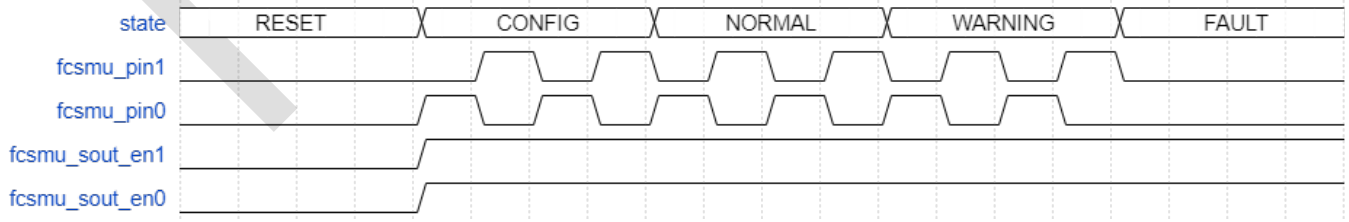

3) TIME-SWITCH(SOCTRL[POLSW] is 0)




4) FAULT-TOGGLE(SOCTRL[POLSW] is 0)




5) TIME-DUAL-RAIL(SOCTRL[POLSW] is 0)



## 39.7 Register Definition

FCSMU base address : 4000_F000h

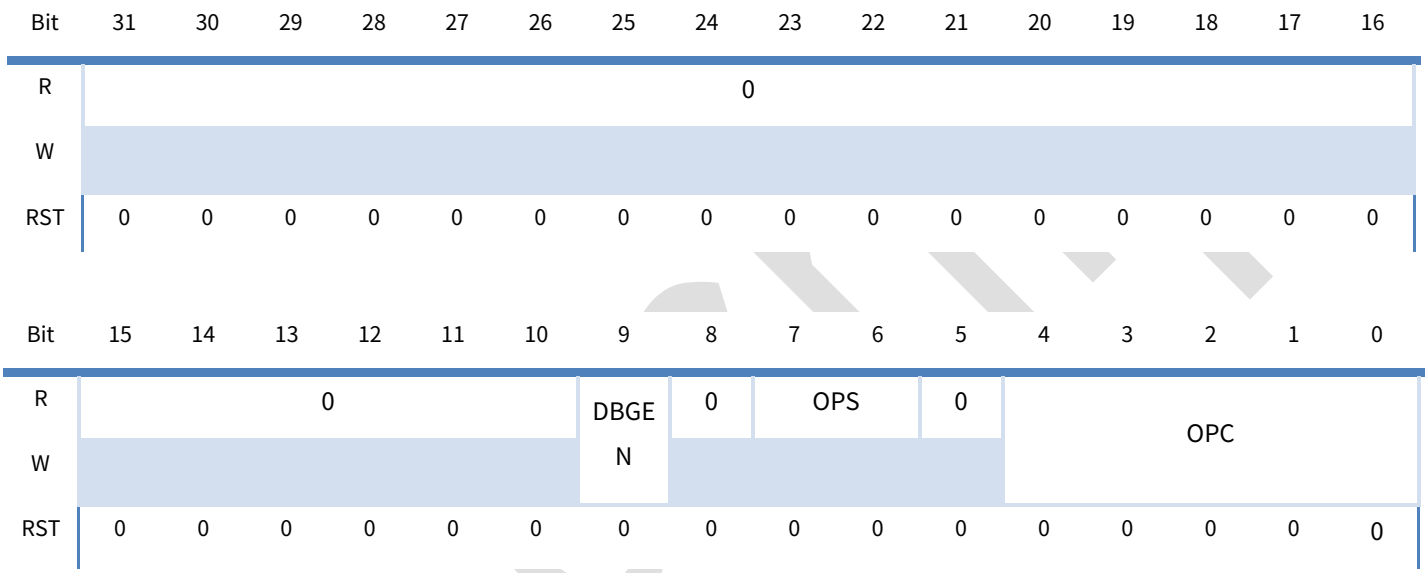
**Register Overview - FCSMU**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | FCSMU_CTRL (Control Register) | 32 | RW | 0000_0000h | page |
| **4h** | FCSMU_OPRK (Operation Key Register) | 32 | WO | 0000_0000h | page |
| **8h** | FCSMU_SOCTRL (Status Output Control Register) | 32 | RW | 0000_0000h | page |
| **Ch** | FCSMU_FCCR0 (Fault Clear Control Register 0) | 32 | RW | FFFF_FFFFh | page |
| **1Ch** | FCSMU_FRST0 (Fault Reset Control Register 0) | 32 | RW | 0000_0000h | page |
| **2Ch** | FCSMU_FST0 (Fault Status Register) | 32 | W1C | 0000_0000h | page |
| **3Ch** | FCSMU_FST_UNLK (Fault Status Register Unlock Register) | 32 | WO | 0000_0000h | page |
| **40h** | FCSMU_FE0 (Fault Enable Register) | 32 | RW | 0000_0000h | page |
| **50h** | FCSMU_WARNING_EN0 (Warning State Enable Register 0) | 32 | RW | FFFF_FFFFh | page |
| **60h** | FCSMU_WARNING_TO (Warning Timeout Interval Register) | 32 | RW | 0003_A980h | page |
| **64h** | FCSMU_CFG_TO (Configuration State Timeout Interval Register) | 32 | RW | 0000_0005h | page |
| **68h** | FCSMU_SOUT_DIAG (SOUT Diagnostic Register) | 32 | RW | 0000_0000h | page |
| **6Ch** | FCSMU_STATUS (Status Register) | 32 | RO | 0000_0010h | page |
| **70h** | FCSMU_NTW (Normal to Warning Register) | 32 | RO | 0000_0000h | page |
| **74h** | FCSMU_WTF (Warning to Fault Register) | 32 | RO | 0000_0000h | page |
| **78h** | FCSMU_NTF (Normal to Fault Register) | 32 | RO | 0000_0000h | page |
| **7Ch** | FCSMU_FTW (Fault to Warning Register) | 32 | RO | 0000_0000h | page |
| **84h** | FCSMU_INJECT (Fault Injection Register) | 32 | WO | 0000_0000h | page |
| **88h** | FCSMU_IRQ_STAT (IRQ Status Register) | 32 | RW | 0000_0000h | page |
| **8Ch** | FCSMU_IRQ_EN (IRQ Enable Register) | 32 | RW | 0000_0000h | page |
| **94h** | FCSMU_TEMP_UNLK (Temporary Configuration State Unlock Register) | 32 | WO | 0000_0000h | page |
| **98h** | FCSMU_PERMNT_LOCK (Permanent Configuration State Lock Register) | 32 | WO | 0000_0000h | page |
| **9Ch** | FCSMU_STMR (SOUT Timer Interval Register) | 32 | RW | 0000_0000h | page |
| **A0h** | FCSMU_WARNING_IEN0 (Warning State Interrupt Enable Register) | 32 | RW | 0000_0000h | page |
| **B0h** | FCSMU_FAULT_IEN0 (Fault State Interrupt Enable Register) | 32 | RW | 0000_0000h | page |
| **C0h** | FCSMU_SOUT_EN0 (SOUT Enable Register) | 32 | RW | 0000_0000h | page |

| | | | | | |
|---|---|---|---|---|---|
| **D0h** | FCSMU_WARNING_TMR (Warning State Timer Register) | 32 | RO | 0003_A980h | page |
| **D4h** | FCSMU_SM_TMR (Safe Mode State Timer Register) | 32 | RO | 0000_0001h | page |
| **D8h** | FCSMU_CFG_TMR (Configuration State Timer Register) | 32 | RO | 000F_FFFFh | page |
| **DCh** | FCSMU_SOUT_TMR (SOUT Timer Register) | 32 | RO | 0000_0000h | page |
| **E0h** | FCSMU_CRC_CTRL (CRC Control Register) | 32 | RW | 0000_0000h | page |
| **E4h** | FCSMU_CRC_RES (CRC Result Register) | 32 | RW | 0000_0000h | page |

### 39.7.1 FCSMU_CTRL (Control Register)

Offset : 0h




**FCSMU_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **9**<br>**DBGEN** | RW | Debug Mode Enable<br>0b - FCSMU is disabled in debug mode.<br>1b - FCSMU is enabled in debug mode. |
| **7-6**<br>**OPS** | RO | Operation Status<br>00b - Idle<br>01b - Busy<br>10b - Fail<br>11b - Success |

| | | |
|---|---|---|
| **4-0**<br>**OPC** | RW | Operation Code<br>00000b - No operation<br>00001b - Move to the configuration state.<br>00010b - Move to the NORMAL state.<br>01101b - Clear fault channel information registers.<br>01111b - Clear OPS to idle.<br>Others - Reserved. |

### 39.7.2 FCSMU_OPRK (Operation Key Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | OPKEY | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | OPKEY | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_OPRK field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**OPKEY** | WO | Operation Unlock Key<br>FC2020CFh - Unlock OPC1.<br>20FCCF20h - Unlock OPC2.<br>Note: The software needs to start the operation immediately after unlocking the operation. |

### 39.7.3 FCSMU_SOCTRL (Status Output Control Register)

Writable only in the configuration state.

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | SOUT_PEN | SOUT_CTRL | | 0 | DIVEX | SMRDT | | | | 0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | FAST EN | POLS W | SOUT_PTC | | | SOUT_DIV | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_SOCTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **25**<br>**SOUT_PEN** | RW | SOUT PINs Enable<br>0b - SOUT PINs is disabled.<br>1b - SOUT PINs is enabled. |
| **24-23**<br>**SOUT_CTRL** | RW | SOUT Control<br>00b - SOUT is controlled by the FSM.<br>01b - SOUT keeps low.<br>10b - SOUT is controlled by the FSM.<br>11b - SOUT keeps high until a fault occurs on a channel, then controlled by FSM. |
| **21**<br>**DIVEX** | RW | SOUT Divider Extend Control<br>0b - SOUT_freq = CLKSAFE_freq/((SOUT_DIV+1)*2*256)<br>1b - SOUT_freq = CLKSAFE_freq/((SOUT_DIV+1)*2*64) |
| **20-17**<br>**SMRDT** | RW | Safe Mode Request Delay Timer<br>Configure the safe mode request delay in cycles of safe clock.<br>0000b - 1 cycle<br>0001b - 2 cycles<br>0010b - 4 cycles<br>0011b - 8 cycles<br>…<br>1111b - 32,768 cycles |
| **14**<br>**FASTEN** | RW | Fast Mode Enable<br>0b - Fast mode is disabled.<br>1b - Fast mod is enabled.<br>Note: Fast mode only applies to Dual-Rail and Time-Switch modes. SOUT will update immediately after FCSMU goes to the fault state. |
| **13**<br>**POLSW** | RW | Polarity Switch<br>0b - Default polarity.<br>1b - Switch polarity. |

| | | |
|---|---|---|
| **12-10**<br>**SOUT_PTC** | RW | SOUT Protocol<br>000b - Dual-Rail<br>001b - Time-Switch<br>010b - Bistable<br>011b - Fault-Toggle<br>100b - Time-Dual-Rail<br>101b - DIAG0<br>110b - DIAG1<br>111b - DIAG2 |
| **9-0**<br>**SOUT_DIV** | RW | SOUT Divider Ratio<br>The divider ratio is controlled by SOUT_DIV and DIVEX.<br>When DIVEX is 0, SOUT_freq = CLKSAFE_freq/((SOUT_DIV+1)*2*256).<br>When DIVEX is 1, SOUT_freq = CLKSAFE_freq/((SOUT_DIV+1)*2*64). |

### 39.7.4 FCSMU_FCCR0 (Fault Clear Control Register 0)

Writable only in the configuration state.

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FCC | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FCC | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**FCSMU_FCCR0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**FCC** | RW | Fault Clear Control<br>FCC[n] is 0b - Fault channel n is cleared by hardware<br>FCC[n] is 1b - Fault channel n is cleared by software |

### 39.7.5 FCSMU_FRST0 (Fault Reset Control Register 0)

Writable only in the configuration state.

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FRST | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FRST | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_FRST0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**FRST** | RW | Fault Reset Control<br>FRST[n] is 0b - Disable reset for channel n.<br>FRST[n] is 1b - Enable reset for channel n. |

### 39.7.6 FCSMU_FST0 (Fault Status Register)

Write FST_UNLK register to unlock before writing FSTx.

Offset : 2Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ST | | | | | | | | | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ST | | | | | | | | | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_FST0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**ST** | W1C | Fault Status<br>ST[n] is 0b - No fault detected for channel n.<br>ST[n] is 1b - Fault is detected for channel n. |

### 39.7.7 FCSMU_FST_UNLK (Fault Status Register Unlock Register)

Write FST_UNLK register to unlock before writing FSTx.

Offset : 3Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | KEY | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | KEY | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_FST_UNLK field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**KEY** | WO | Key<br>Write a key number to this register to unlock FSTx register, so that you can clear the fault status.<br>951413CFh - Unlock fault status registers<br>Others - Nothing |

### 39.7.8 FCSMU_FE0 (Fault Enable Register)

Writable only in the configuration state.

Offset : 40h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EN | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EN | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_FE0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**EN** | RW | Enable Bits<br>EN[n] is 0b - Fault channel n is disabled.<br>EN[n] is 1b - Fault channel n is enabled. |

### 39.7.9 FCSMU_WARNING_EN0 (Warning State Enable Register 0)

Writable only in the configuration state.

Offset : 50h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EN | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EN | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**FCSMU_WARNING_EN0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**EN** | RW | Enable Bits<br>EN[n] is 0b - Warning state is disabled for fault channel n.<br>EN[n] is 1b - Warning state is enabled for fault channel n. |

### 39.7.10 FCSMU_WARNING_TO (Warning Timeout Interval Register)

Writable only in the configuration state.

Offset : 60h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TIME | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIME | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_WARNING_TO field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**TIME** | RW | Warning Timeout Interval<br>This field controls the maximum amount of time that FCSMU can be in the warning state in cycles of safe clock.<br>Tmax = TIME cycles of safe clock. |

### 39.7.11 FCSMU_CFG_TO (Configuration State Timeout Interval Register)

Writable only in normal, warning and fault states.

Offset : 64h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | TIME | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

**FCSMU_CFG_TO field descriptions**

| Field | Type | Description |
|---|---|---|
| **2-0**<br>**TIME** | RW | Configuration State Timeout Interval<br>This field controls the maximum amount of time that FCSMU can be in the configuration state in cycles of safe clock.<br>Tmax = 2(TIME+11) cycles |

### 39.7.12 FCSMU_SOUT_DIAG (SOUT Diagnostic Register)

Offset : 68h

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

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | SIN1 | SIN0 | 0 | | SOUT 1 | SOUT 0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_SOUT_DIAG field descriptions**

| Field | Type | Description |
|---|---|---|
| **5**<br>**SIN1** | RO | SOUT Input 1<br>0b - SIN1 is low<br>1b - SIN1 is high<br>Note: Set SOCTRL[SOUT_PEN] when using this field. |
| **4**<br>**SIN0** | RO | SOUT Input 0<br>0b - SIN0 is low<br>1b - SIN0 is high<br>Note: Set SOCTRL[SOUT_PEN] when using this field. |
| **1**<br>**SOUT1** | RW | SOUT Output 1<br>0b - Force SOUT1 low in DIAG0/1 mode.<br>1b - Force SOUT1 high in DIAG0/1 mode. |
| **0**<br>**SOUT0** | RW | SOUT Output 0<br>0b - Force SOUT0 low in DIAG1/2 mode.<br>1b - Force SOUT0 high in DIAG1/2 mode. |

### 39.7.13 FCSMU_STATUS (Status Register)

Offset : 6Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | SOUTPIN | FIF | STAT | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**FCSMU_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **5-4**<br>**SOUTPIN** | RO | SOUT PINs<br>00b - SOUT = 2'b00<br>01b - SOUT = 2'b01<br>10b - SOUT = 2'b10<br>11b - SOUT = 2'b11 |
| **3**<br>**FIF** | RO | FCSMU In Fault<br>0b - FCSMU is not in fault state.<br>1b - FCSMU is in fault state. |
| **2-0**<br>**STAT** | RO | FCSMU State<br>000b - Normal<br>001b - Configuration<br>010b - Warning<br>011b - Fault |

### 39.7.14 FCSMU_NTW (Normal to Warning Register)

Offset : 70h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | INDEX | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_NTW field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **7-0**<br>**INDEX** | RO | Fault Channel Index<br>This field indicates which fault channel causes FCSMU moves from the normal state to the warning state.<br>00h - No fault.<br>01h - CH0<br>02h - CH1<br>03h - CH2<br>…<br>80h - CH127<br>FFh - Multiple faults<br>Others - Reserved |

### 39.7.15 FCSMU_WTF (Warning to Fault Register)

Offset : 74h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FLAG | 0 | INDEX | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_WTF field descriptions**

| Field | Type | Description |
|---|---|---|
| **9**<br>**FLAG** | RO | Warning to Fault State Flag<br>0b - No warning to fault transition.<br>1b - Warning to fault transition. |

| | | |
|---|---|---|
| **7-0**<br>**INDEX** | RO | Fault Index<br>This field indicates which fault channel causes FCSMU moves from the warning state to the fault state.<br>00h - No fault.<br>01h - CH0<br>02h - CH1<br>03h - CH2<br>…<br>80h - CH127<br>FFh - Multiple faults<br>Others - Reserved |

### 39.7.16 FCSMU_NTF (Normal to Fault Register)

Offset : 78h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | FLAG | 0 | INDEX | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_NTF field descriptions**

| Field | Type | Description |
|---|---|---|
| **9**<br>**FLAG** | RO | Normal to Fault State Flag<br>0b - No normal to fault transition.<br>1b - Normal to fault transition. |

| | | |
|---|---|---|
| **7-0**<br>**INDEX** | RO | Fault Index<br>This field indicates which fault channel causes FCSMU moves from the normal state to the fault state.<br>00h - No fault.<br>01h - CH0<br>02h - CH1<br>03h - CH2<br>…<br>80h - CH127<br>FFh - Multiple faults<br>Others - Reserved |

### 39.7.17 FCSMU_FTW (Fault to Warning Register)

Offset : 7Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | INDEX | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_FTW field descriptions**

| Field | Type | Description |
|---|---|---|
| **7-0**<br>**INDEX** | RO | Fault Index<br>This field indicates which fault channel causes FCSMU moves from the fault state to the warning state.<br>00h - No fault.<br>01h - CH0<br>02h - CH1<br>03h - CH2<br>…<br>80h - CH127<br>FFh - Multiple faults<br>Others - Reserved |

### 39.7.18 FCSMU_INJECT (Fault Injection Register)

Offset : 84h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | 0 | | | | | | |
| W | | | | | | | | | | INDEX | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_INJECT field descriptions**

| Field | Type | Description |
|---|---|---|
| **6-0**<br>**INDEX** | WO | Fault Injection Index<br>Inject a fault on a specified channel. The fault can be cleared by writing FST0~FST3.<br>00h - Inject fault on CH0<br>01h - Inject fault on CH1<br>…<br>7Fh - Inject fault on CH127<br>Others - Reserved |

### 39.7.19 FCSMU_IRQ_STAT (IRQ Status Register)

Offset : 88h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | FAULT_IRQ | WARNING_IRQ | CFG_TO_IRQ |
| W | | | | | | | | | | | | | | | | w1c |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_IRQ_STAT field descriptions**

| Field | Type | Description |
|---|---|---|
| **2**<br>**FAULT_IRQ** | RO | Fault Interrupt Request<br>0b - Fault interrupt request disassert<br>1b - Fault interrupt request assert |
| **1**<br>**WARNING_IRQ** | RO | Warning Interrupt Request<br>0b - Warning interrupt request disassert<br>1b - Warning interrupt request assert |
| **0**<br>**CFG_TO_IRQ** | W1C | Configuration Timeout Interrupt Request<br>0b - Configuration timeout interrupt request disassert<br>1b - Configuration timeout interrupt request assert |

### 39.7.20 FCSMU_IRQ_EN (IRQ Enable Register)

Offset : 8Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | CFG_TO_IEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_IRQ_EN field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**CFG_TO_IEN** | RW | Configuration Timeout Interrupt Enable<br>0b - Configuration timeout interrupt is disabled.<br>1b - Configuration timeout interrupt is enabled. |

### 39.7.21 FCSMU_TEMP_UNLK (Temporary Configuration State Unlock Register)

Writable only in supervisor mode.

Offset : 94h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | 0 | | | | | | | | |
| W | | | | | | | | KEY | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_TEMP_UNLK field descriptions**

| Field | Type | Description |
|---|---|---|
| **8-0**<br>**KEY** | WO | Temporary Configuration State Unlock Key<br>FCh - Unlock the configuration state temporarily.<br>Others - Lock the configuration state temporarily. |

### 39.7.22 FCSMU_PERMNT_LOCK (Permanent Configuration State Lock Register)

Writable only in supervisor mode.

Offset : 98h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | 0 | | | | | | | | |
| W | | | | | | | | KEY | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_PERMNT_LOCK field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **8-0**<br>**KEY** | WO | Permanent Configuration State Lock Key<br>CFh - Lock the configuration state permanently.<br>Others - Nothing. |

### 39.7.23 FCSMU_STMR (SOUT Timer Interval Register)

Writable only in the configuration state.

Offset : 9Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FTTI | | 0 | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | MTE | MINI_TIME | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_STMR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-30**<br>**FTTI** | RW | Fault Tolerance Time<br>If FCSMU fails to react to a fault within FTTI time, the chip will be reset.<br>00b - 1ms<br>01b - 2ms<br>10b - 5ms<br>11b - 10ms |
| **14**<br>**MTE** | RW | SOUT Minimum Timer Enable<br>0b - Tmin only work for FAULT-TOGGLE mode. The SOUT changes from fault status to non-fault immediately after FCSMU exits FAULT state.<br>1b - When FCSMU exits FAULT state, SOUT keeps stable in Fault-Toggle, BISTABLE and TIMESWITCH mode for Tmin. |
| **13-0**<br>**MINI_TIME** | RW | Minimum SOUT Timer Interval<br>This field controls the minimum amount of time that SOUT keeps stable in Fault-Toggle, BISTABLE and TIMESWITCH mode.<br>Tmin = 250us + MINI_TIME |

### 39.7.24 FCSMU_WARNING_IEN0 (Warning State Interrupt Enable Register)

Writable only in the configuration state.

Offset : A0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EN | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EN | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_WARNING_IEN0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**EN** | RW | Enable Bits<br>EN[n] is 0b - Warning interrupt is disabled for fault channel n.<br>EN[n] is 1b - Warning interrupt is enabled for fault channel n. |

### 39.7.25 FCSMU_FAULT_IEN0 (Fault State Interrupt Enable Register)

Writable only in the configuration state.

Offset : B0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EN | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EN | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_FAULT_IEN0 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-0**<br>**EN** | RW | Enable Bits<br>EN[n] is 0b - Fault interrupt is disabled for fault channel n.<br>EN[n] is 1b - Fault interrupt is enabled for fault channel n. |

### 39.7.26 FCSMU_SOUT_EN0 (SOUT Enable Register)

Writable only in the configuration state.

Offset : C0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EN | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EN | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_SOUT_EN0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**EN** | RW | Enable Bits<br>EN[n] is 0b - SOUT is disabled for fault channel n.<br>EN[n] is 1b - SOUT is enabled for fault channel n. |

### 39.7.27 FCSMU_WARNING_TMR (Warning State Timer Register)

Offset : D0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_WARNING_TMR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**VAL** | RO | Warning State Timer value<br>This field indicates the value of the warning state timer. |

### 39.7.28 FCSMU_SM_TMR (Safe Mode State Timer Register)

Offset : D4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**FCSMU_SM_TMR field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**VAL** | RO | Safe Mode Timer Value<br>This field indicates the value of the safe mode timer. |

### 39.7.29 FCSMU_CFG_TMR (Configuration State Timer Register)

Offset : D8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | VAL | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**FCSMU_CFG_TMR field descriptions**

| Field | Type | Description |
|---|---|---|
| **20-0**<br>**VAL** | RO | Configuration State Timer Value<br>This field indicates the value of the configuration state timer. |

### 39.7.30 FCSMU_SOUT_TMR (SOUT Timer Register)

Offset : DCh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | VAL | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_SOUT_TMR field descriptions**

| Field | Type | Description |
|---|---|---|
| **22-0**<br>**VAL** | RO | SOUT Timer value<br>This field indicates the value of the SOUT timer. |

### 39.7.31 FCSMU_CRC_CTRL (CRC Control Register)

Offset : E0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | DONE | EF | BUSY | EOEN | CHKEN | TRGEN | 0 |
| W | | | | | | | | | | w1c | w1c | | | | | GEN |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_CRC_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **6**<br>**DONE** | W1C | CRC Done Flag<br>0b - CRC not done.<br>1b - CRC done. |
| **5**<br>**EF** | W1C | CRC Error Flag<br>0b - No CRC error.<br>1b - CRC error is detected. |
| **4**<br>**BUSY** | RO | CRC Busy Indicator<br>0b - CRC generator is idle.<br>1b - CRC generator is busy. |
| **3**<br>**EOEN** | RW | Error Output Enable<br>0b - CRC error output disable.<br>1b - CRC error output enable. |
| **2**<br>**CHKEN** | RW | CRC Checker Enable<br>0b - Disable CRC checker.<br>1b - Enable CRC checker. |
| **1**<br>**TRGEN** | RW | Trigger Enable<br>0b - Disable trigger.<br>1b - Enable trigger, then trigger input can start CRC generation. |
| **0**<br>**GEN** | WO | CRC Generate<br>0b - Do nothing.<br>1b - Start a CRC generate operation. |

### 39.7.32 FCSMU_CRC_RES (CRC Result Register)

Offset : E4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RESULT | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RESULT | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSMU_CRC_RES field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**RESULT** | RW | CRC Result Value<br>This field stores the CRC result generated.<br>When CRC_CTRL[CHKEN] is set, reading this register gets the CRC result. It's a read-only register.<br>When CRC_CTRL[CHKEN] is cleared, writing this register can configure the golden CRC value. Reading this register can get the golden CRC value. |

Flagchip

# Chapter 40 Safety Test and Control Unit (STCU)

## 40.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • Changed the name of the bit field "PULPINO" to "SUBSYS" in the register sections M *STCU_BIST_SEL, STCU_MBIST_DONE_STATUS,* and *STCU_MBIST_FAIL_STATUS;*<br>• Updated the description of the "MODE" bit field in the register section *STCU_SRAM_INI_CTRL* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 40.2 Chip-specific Information

N/A.

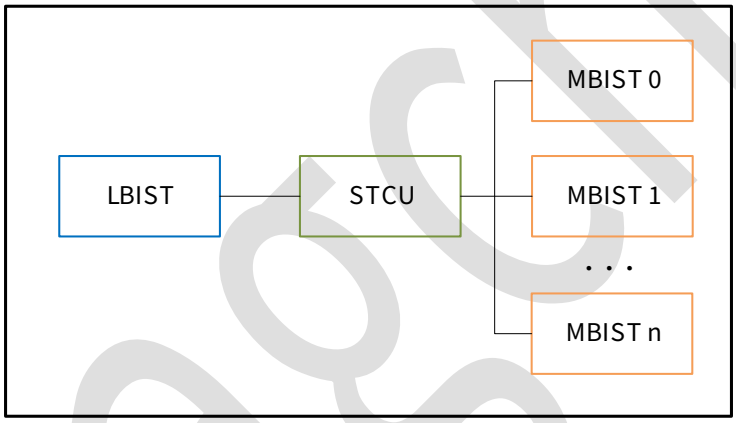
## 40.3 Features

- Power-on self-test.
- Software triggered self-test.
- Full chip Logic Built-In Self-Test (LBIST).
- MBIST with all SRAMs and ROMs covered.
- Hardware SRAM initialization.
- Internal watchdog for self-test timeout detection.

## 40.4 Overview

The Safety Test and Control Unit (STCU) is to control offline self-test sequences including LBIST and Memory Built-In Self-Test (MBIST). There are one LBIST engine and lots of MBIST engines inside the chip, the STCU directly controls these engines to complete self-test. The STCU also provides a hardware initialization function for DTCM, ITCM, and system SRAM.

**Figure 66. Self-test structure**




## 40.5 Functional Description

### 40.5.1 Power-on Self-test

When the chip powers on, self-test configuration data in flash NVR will be loaded into STCU control registers if the NVR safety codes are correct. Self-test will be triggered if the below requirements are both satisfied after the NVR load.

1) Self-test is not bypassed (STCU_SELF_TEST_CTRL.STEST_BYPASS is 0).
2) At least one BIST is enabled (STCU_SELF_TEST_CTRL.LBIST_EN or STCU_SELF_TEST_CTRL.MBIST_EN is 1).

The self-test result can be checked by accessing the STCU status registers after reset release.

### 40.5.2 Software Self-test

To trigger self-test by software, the user needs to perform the below sequence:

1) Clear the STCU status registers: STCU_SELF_TEST_STATUS/STCU_LBIST_STATUS/STCU_MBIST_DONE_STATUS/STCU_MBIST_FAIL_STATUS.
2) Set up self-test configuration by writing STCU control registers. This step can be omitted if these registers already have NVR loaded values and don’ t need to change. After setup, the below requirements should be satisfied.
   a) Self-test is not bypassed (STCU_SELF_TEST_CTRL.STEST_BYPASS is 0).
   b) At least one BIST is enabled (STCU_SELF_TEST_CTRL.LBIST_EN or STCU_SELF_TEST_CTRL.MBIST_EN is 1).

c) Due to MBIST algorithm differences, the setting for maximum timeout time may need to be larger than the power on self-test setting.

d) STCU_SELF_TEST_RESET.RST should be 0.

3) Write 32'hBB40_E64D to STCU_SELF_TEST_KEY.
4) Set the TRIG bits in both STCU_SELF_TEST_TRIG_A and STCU_SELF_TEST_TRIG_B.
5) Wait self-test done, the chip goes into the reset state.
6) Self-test done, chip reset release.
7) Check STCU status registers.

### 40.5.3 Chip Status during Self-test

When the self-test is under execution, the chip will go into a reset state and become unavailable, all ports (except RESET_B and JTAG ports) can be configured to HIZ, pull-up or pull-down state. Also, the self-test status can be observed on FCSMU_PIN0 and FCSMU_PIN1 as the two figures below when dedicated FCSMU pins are enabled and STCU_SELF_TEST_CTRL.STEST_ST is 0. If STCU_SELF_TEST_CTRL.STEST_ST is 1 and dedicated FCSMU pins are enabled, the value of bit FCSMU_PIN0_VAL and FCSMU_PIN1_VAL will be output on FCSMU pins during the self-test.

**Figure 67. FCSMU_PIN status when enabling both two pins**

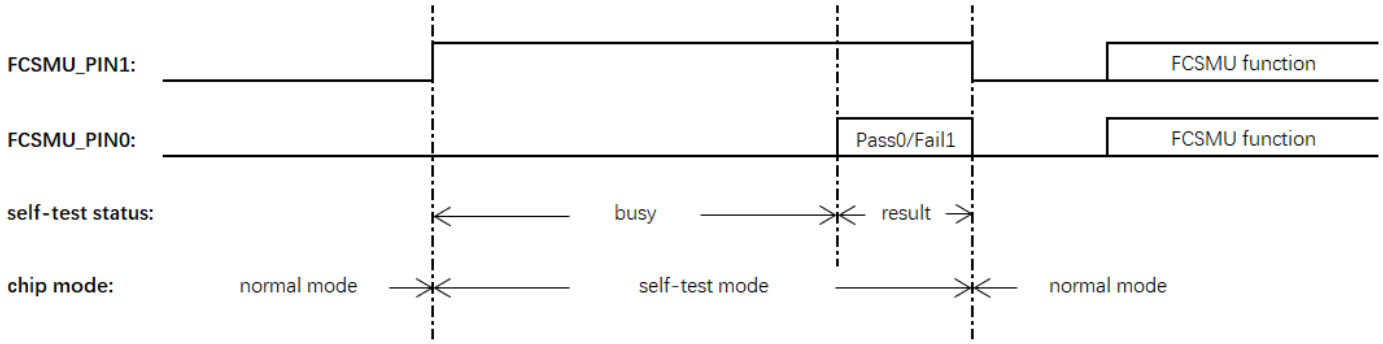
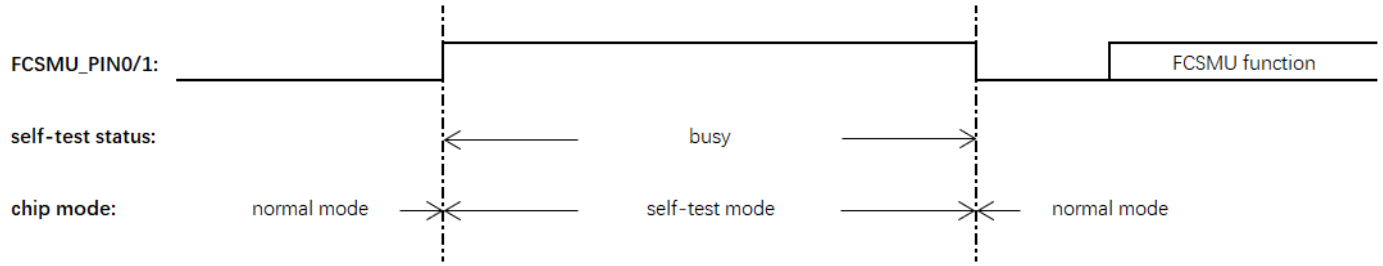



**Figure 68. FCSMU_PIN status when enabling only one pin**




### 40.5.4 Self-Test Sequence

Both LBIST and MBIST can be enabled independently for self-test. When they are both enabled, LBIST will always be executed first. When both are disabled, the self-test will be bypassed if triggered.

### 40.5.5 Pin Reset Control

When the self-test is under execution, de-assert RESET_B (1 -> 0) will abort the execution, the self-test will end with the abort flag asserted. When the chip is in normal mode, de-assert RESET_B (1-> 0) can also trigger a self-test if the control bit STCU_SELF_TEST_CTRL.RST_TRIG is 0.

### 40.5.6 FCSMU Reset Control

When an FCSMU reset happens, a self-test can also be triggered if the control bit SELF_TEST_CTRL[FCSMU_TRIG] is 0.

### 40.5.7 Watchdog Timer

There is a programmable watchdog timer inside the STCU, which will start counting when a self-test is triggered. If the self-test execution time exceeds the maximum time set in STCU_SELF_TEST_CTRL.MT, the self-test will be terminated with a timeout error.

### 40.5.8 STCU Self-check

To enhance the function safety of the STCU itself, if a self-test is triggered with STCU_SELF_TEST_CTRL.SCHK_EN asserted, the STCU will run a self-check before the self-test starts. This self-check targets faults in the STCU, and if this check fails, the self-test will not be executed.

### 40.5.9 NVR Data Load Check

When powering on, self-test configuration data in the flash NVR will be loaded into STCU control registers of the table below to support power on test. To enhance function safety, each 64-bit NVR data has 4 safety bits. Only if these safety bits are programmed with the correct code as the table below will the NVR data be loaded into the corresponding STCU register. If wrong safety bits are detected, these STCU registers will keep the reset value and an error flag will be set. Besides the 4 bits, there are some other codes listed in the NVR Safety Code column of the below table which also need to be programmed to make self-test work.

**Table 65. NVR data load of STCU**

| NVR Configuration Name | BIT | Description | STCU Register | NVR Safety Code |
|---|---|---|---|---|
| Self-test Ctrl | [63:60] | - | - | 0xA |
| | [59:32] | self_test_ctrl1 | - | - |
| | [31:0] | self_test_ctrl0 | STCU_SELF_TEST_CTRL | - |
| MBIST Ctrl | [63:60] | - | - | 0x5 |
| | [59:32] | mbist_alg | STCU_MBIST_ALG[27:0] | - |
| | [31:0] | mbist_sel | STCU_MBIST_SEL | - |
| LBIST Ctrl | [63:60] | | - | 0x6 |
| | [59:32] | lbist_ctrl1 | - | 0x0084FBD |
| | [31:0] | lbist_ctrl0 | STCU_LBIST_PAT_CTRL | - |
| LBIST Misr | [63:60] | | - | 0x9 |
| | [59:32] | lbist_misr1 | - | - |
| | [31:0] | lbist_misr0 | STCU_LBIST_EXP_MISR | - |

### 40.5.10 Clock Source

A self-test can select PLL or FIRC as the clock source, it is recommended to select PLL for at-speed test coverage and less test time.

### 40.5.11 LBIST

The LBIST covers the digital logic of the whole chip except the below modules: PMC, SCG, RGM, and STCU. The LBIST pattern amount can be controlled by STCU_LBIST_PAT_CTRL.N. More LBIST patterns will achieve better test coverage while needing more test time. It is advised to set it to recommended value to get a minimum test time with > 90% test coverage.

### 40.5.12 MBIST

All SRAMs and ROMs are separated into several groups which have independent enable control and test result flag. There are two algorithm selections, basic test, and full test. The basic test detects basic faults and requires less test time. The full test detects almost all fault types but requires much more test time. ITCM/DTCM/system SRAM can be initialized at the end of

MBIST if the initialization control is 1. By setting STCU_SELF_TEST_CTRL.MBIST_LP, the MBIST can enable a low-power feature that can reduce power consumption while increasing test time.

### 40.5.13 Interruption

Incorrect register configuration will lead to interruption in the STCU.

- Illegal register access size. The STCU only supports word access.
- Illegal register access sequence, such as software trigger self-test without the correct key or changing SRAM initialization mode when initialization is running.

### 40.5.14 Hardware SRAM Initialization

The STCU provides an online SRAM initialization function for all ITCM/DTCM/system SRAMs. The SRAMs can be selected by groups independently. When initializing SRAM after standby mode, STCU_SRAM_INI_CTRL.MODE should be configured for different standby modes. During the initialization, it is not allowed to access the SRAMs that are being initialized. After the initialization is done, the done flag will be set, and it is recommended to clear the STCU_SRAM_INI_SEL register and lock the initial enable for better function safety.

## 40.6 Register Definition

STCU base address : 4007_F000h

**Register Overview - STCU**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | STCU_SELF_TEST_KEY (Self Test Key Register) | 32 | RW | 0000_0000h | page |
| **4h** | STCU_SELF_TEST_CTRL (Self Test Control Register) | 32 | RW | 2019_C3FFh | page |
| **8h** | STCU_SELF_TEST_STATUS (Self Test Status Register) | 32 | W0C | 0000_0000h | page |
| **Ch** | STCU_SELF_TEST_TRIG_A (Self Test Trigger A Register) | 32 | RW | 0000_0000h | page |
| **10h** | STCU_SELF_TEST_TRIG_B (Self Test Trigger B Register) | 32 | RW | 0000_0000h | page |
| **14h** | STCU_SELF_TEST_RESET (Self Test Reset Register) | 32 | RW | 0000_0000h | page |
| **18h** | STCU_LBIST_PAT_CTRL (LBIST Pattern Control Register) | 32 | RW | 0000_0000h | page |
| **20h** | STCU_LBIST_EXP_MISR (LBIST Expected MISR Register) | 32 | RW | 0000_0000h | page |
| **24h** | STCU_LBIST_ACT_MISR (LBIST Actual MISR Register) | 32 | RO | 0000_0000h | page |
| **28h** | STCU_LBIST_STATUS (LBIST Status Register) | 32 | W0C | 0000_0000h | page |
| **34h** | STCU_MBIST_SEL (MBIST Select Register) | 32 | RW | 0000_0000h | page |
| **38h** | STCU_MBIST_ALG (MBIST Algorithm Register) | 32 | RW | 0000_0000h | page |
| **3Ch** | STCU_MBIST_DONE_STATUS (MBIST Done Status Register) | 32 | W0C | 0000_0000h | page |
| **40h** | STCU_MBIST_FAIL_STATUS (MBIST Fail Status Register) | 32 | W0C | 0000_0000h | page |
| **44h** | STCU_USER_GP (User General Purpose Register) | 32 | RW | 0000_0000h | page |
| **48h** | STCU_SRAM_INI_CTRL (SRAM Initialization Control Register) | 32 | RW | 0000_0000h | page |

| | | | | | |
|---|---|---|---|---|---|
| **4Ch** | STCU_SRAM_INI_STATUS (SRAM Initialization Status Register) | 32 | RO | 0000_0000h | page |
| **50h** | STCU_SRAM_INI_SEL (SRAM Initialization Select Register) | 32 | RW | 0000_0000h | page |
| **54h** | STCU_SRAM_INI_DONE_STATUS (SRAM Initialization Done Status Register) | 32 | RW | 0000_0000h | page |
| **58h** | STCU_IRQ (Interrupt Request Register) | 32 | RW | 0000_0000h | page |

### 40.6.1 STCU_SELF_TEST_KEY (Self Test Key Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | KEY | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | KEY | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_SELF_TEST_KEY field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**KEY** | RW | Safety Key<br>Written to 32'hBB40_E64D will enable STCU_SELF_TEST_CTRL A/B access, cleared when self-test is triggered. |

### 40.6.2 STCU_SELF_TEST_CTRL (Self Test Control Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LBIST_CLKDIV | | FCSMU_TRIG | LBIST_EN | LBIST_TD | SCHK_EN | MBIST_LP | MBIST_EN | 0 | | CLK_SEL | RST_TRIG | STEST_BYPASS | PS | PE | STEST_ST |
| RST | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FCSMU_PIN1_EN | FCSMU_PIN0_EN | FCSMU_PIN1_VAL | FCSMU_PIN0_VAL | MT | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**STCU_SELF_TEST_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-30**<br>**LBIST_CLKDIV** | RW | LBIST Clock Divider Control<br>A slower LBIST clock will cause less power consumption but a longer test time.<br>00b - Divided by 2<br>01b - Divided by 3<br>10b - Divided by 4 (recommended)<br>11b - Divided by 5 |
| **29**<br>**FCSMU_TRIG** | RW | Control if FCSMU reset will trigger self-test<br>0b - FCSMU reset will trigger self-test.<br>1b - FCSMU reset will not trigger self-test. |
| **28**<br>**LBIST_EN** | RW | LBIST Enable<br>0b - Disable LBIST.<br>1b - Enable LBIST. |
| **27**<br>**LBIST_TD** | RW | LBIST At-speed Test Enable<br>0b - Disable LBIST at-speed test.<br>1b - Enable LBIST at-speed test. (recommended) |
| **26**<br>**SCHK_EN** | RW | STCU Self check Enable<br>0b - Disable STCU self check.<br>1b - Enable STCU self check. (recommended) |
| **25**<br>**MBIST_LP** | RW | MBIST Low Power Control Enable<br>0b - Disable MBIST low power control. (recommended)<br>1b - Enable MBIST low power control. |
| **24**<br>**MBIST_EN** | RW | MBIST Enable<br>0b - Disable MBIST.<br>1b - Enable MBIST. |
| **21**<br>**CLK_SEL** | RW | Select the clock source of self-test.<br>0b - FIRC.<br>1b - PLL. (recommended) |

| | | |
|---|---|---|
| **20**<br>**RST_TRIG** | RO | Control if the pin reset will trigger self-test.<br>0b - Pin reset will trigger self-test.<br>1b - Pin reset will not trigger self-test. |
| **19**<br>**STEST_BYPASS** | RW | Self-test Bypass Control<br>0b - Self-test will not be bypassed.<br>1b - Self-test will be bypassed. |
| **18**<br>**PS** | RW | Self-test Port Pull Selection<br>0b - Ports will be pulled down during self-test.<br>1b - Ports will be pulled up during self-test. |
| **17**<br>**PE** | RW | Self-test Port Pull Enable<br>0b - Disable port pull function during self-test.<br>1b - Enable port pull function during self-test. |
| **16**<br>**STEST_ST** | RO | Enable self-test status monitor on FCSMU pins during self-test when the FCSMU pins are enabled.<br>0b - Self-test status will be output on FCSMU pins.<br>1b - Self-test status will not be output on FCSMU pins. |
| **15**<br>**FCSMU_PIN1_EN** | RO | Enable dedicated FCSMU_PIN1 for both self-test and normal mode.<br>0b - Dedicated FCSMU_PIN1 is enabled.<br>1b - Dedicated FCSMU_PIN1 is disabled. |
| **14**<br>**FCSMU_PIN0_EN** | RO | Enable dedicated FCSMU_PIN0 for both self-test and normal mode.<br>0b - Dedicated FCSMU_PIN0 is enabled.<br>1b - Dedicated FCSMU_PIN0 is disabled. |
| **13**<br>**FCSMU_PIN1_VAL** | RO | FCSMU_PIN1 default value during self-test when self-test status monitor is disabled. |
| **12**<br>**FCSMU_PIN0_VAL** | RO | FCSMU_PIN0 default value during self-test when self-test status monitor is disabled. |
| **11-0**<br>**MT** | RW | The maximum time allowed for self-test execution. Self-test will be terminated if execution time exceeds this value. (unit : ms)<br>With recommended self-test control setting, advise 0x010 for power on self-test, 0x040 for software self-test. |

### 40.6.3 STCU_SELF_TEST_STATUS (Self Test Status Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | SCHK_ERR | NVR_LD_ERR | 0 | TIMEOUT_ERR | MBIST_ERR | LBIST_ERR | 0 | | STEST_ABORT | STEST_DONE |
| W | | | | | | | w0c | w0c | | w0c | w0c | w0c | | | w0c | w0c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_SELF_TEST_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **9**<br>**SCHK_ERR** | W0C | STCU Self Check Error Flag<br>0b - No error occurs.<br>1b - Error occurs. |
| **8**<br>**NVR_LD_ERR** | W0C | NVR Load Error Flag<br>0b - No error occurs.<br>1b - Error occurs. |
| **6**<br>**TIMEOUT_ERR** | W0C | Self-test Timeout Error Flag<br>0b - No error occurs.<br>1b - Error occurs. |
| **5**<br>**MBIST_ERR** | W0C | MBIST Error Flag<br>0b - No error occurs.<br>1b - Error occurs. |
| **4**<br>**LBIST_ERR** | W0C | LBIST Error Flag<br>0b - No error occurs.<br>1b - Error occurs. |
| **1**<br>**STEST_ABORT** | W0C | Self-test Abort Flag<br>0b - No self-test abort event occurs.<br>1b - Self-test abort event occurs. |
| **0**<br>**STEST_DONE** | W0C | Self-test Done Flag<br>0b - No self-test done event occurs.<br>1b - Self-test done event occurs. |

### 40.6.4 STCU_SELF_TEST_TRIG_A (Self Test Trigger A Register)

Offset : Ch

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

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | TRIG |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_SELF_TEST_TRIG_A field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**TRIG** | RW | Self-test Software Trigger Enable<br>Only both TRIG bits in STCU_SELF_TEST_TRIG_A/B are enabled will self-test be triggered, self clear by trigger.<br>0b - Disable self-test software trigger.<br>1b - Enable self-test software trigger. |

### 40.6.5 STCU_SELF_TEST_TRIG_B (Self Test Trigger B Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | TRIG |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_SELF_TEST_TRIG_B field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**TRIG** | RW | Self-test Software Trigger Enable<br>Only both TRIG bits in STCU_SELF_TEST_TRIG_A/B are enabled will self-test be triggered, self clear by trigger.<br>0b - Disable self-test software trigger.<br>1b - Enable self-test software trigger. |

### 40.6.6 STCU_SELF_TEST_RESET (Self Test Reset Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | RST |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_SELF_TEST_RESET field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**RST** | RW | Assert this bit will put self-test logic under reset for function safety.<br>0b - Self-test logic is not reset.<br>1b - Self-test logic is reset. |

### 40.6.7 STCU_LBIST_PAT_CTRL (LBIST Pattern Control Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | N | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_LBIST_PAT_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **9-0**<br>**N** | RW | LBIST Pattern Amount Control Value N<br>Pattern amount = N * 64, 0xB is recommended. |

### 40.6.8 STCU_LBIST_EXP_MISR (LBIST Expected MISR Register)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MISR | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MISR | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_LBIST_EXP_MISR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**MISR** | RW | LBIST Expected MISR Signature<br>With recommended LBIST setting, should be 0x00AFB99B. |

### 40.6.9 STCU_LBIST_ACT_MISR (LBIST Actual MISR Register)

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MISR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MISR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_LBIST_ACT_MISR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-0**<br>**MISR** | RO | LBIST Actual MISR Signature |

### 40.6.10 STCU_LBIST_STATUS (LBIST Status Register)

Offset : 28h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | FAIL |
| W | | | | | | | | | | | | | | | | w0c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | DONE |
| W | | | | | | | | | | | | | | | | w0c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_LBIST_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **16**<br>**FAIL** | W0C | LBIST Fail Flag<br>0b - No LBIST fail event occurs.<br>1b - LBIST fail event occurs. |
| **0**<br>**DONE** | W0C | LBIST Done Flag.<br>0b - No LBIST done event occurs.<br>1b - LBIST done event occurs. |

### 40.6.11 STCU_MBIST_SEL (MBIST Select Register)

Offset : 34h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | HSM_ROM | SYS_ROM | DMA_FLEXCAN | HSM | SUBSYS | CACHE_CPU0 | DTCM1_CPU0 | DTCM0_CPU0 | ITCM_CPU0 | SRAM |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_MBIST_SEL field descriptions**

| Field | Type | Description |
|---|---|---|
| **9**<br>**HSM_ROM** | RW | HSM ROM Select for MBIST<br>0b - Not select.<br>1b - Select. |
| **8**<br>**SYS_ROM** | RW | System ROM Select for MBIST<br>0b - Not select.<br>1b - Select. |
| **7**<br>**DMA_FLEXCAN** | RW | DMA_FLEXCAN SRAM Select for MBIST<br>0b - Not select.<br>1b - Select. |
| **6**<br>**HSM** | RW | HSM SRAM Select for MBIST<br>0b - Not select.<br>1b - Select. |
| **5**<br>**SUBSYS** | RW | HSM/FlexCore Subsystem SRAM Select for MBIST<br>0b - Not select.<br>1b - Select. |
| **4**<br>**CACHE_CPU0** | RW | CACHE_CPU0 SRAM Select for MBIST<br>0b - Not select.<br>1b - Select. |
| **3**<br>**DTCM1_CPU0** | RW | DTCM1_CPU0 SRAM Select for MBIST<br>0b - Not select.<br>1b - Select. |
| **2**<br>**DTCM0_CPU0** | RW | DTCM0_CPU0 SRAM Select for MBIST<br>0b - Not select.<br>1b - Select. |
| **1**<br>**ITCM_CPU0** | RW | ITCM_CPU0 SRAM Select for MBIST<br>0b - Not select.<br>1b - Select. |
| **0**<br>**SRAM** | RW | System SRAM Select for MBIST<br>0b - Not select.<br>1b - Select. |

### 40.6.12 STCU_MBIST_ALG (MBIST Algorithm Register)

Offset : 38h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

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

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | TRIG_INI_EN | TRIG_ALG_SEL | | | | | 0 | | POR_INI_EN | POR_ALG_SEL | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_MBIST_ALG field descriptions**

| Field | Type | Description |
|---|---|---|
| **13**<br>**TRIG_INI_EN** | RW | Enable SRAM initialization at the end of software trigger self-test.<br>0b - Disable SRAM initialization.<br>1b - Enable SRAM initialization. (recommended) |
| **12-8**<br>**TRIG_ALG_SEL** | RW | MBIST algorithm select for software trigger self-test.<br>11001b - Basic test.<br>11111b - Full test. (recommended) |
| **5**<br>**POR_INI_EN** | RW | Enable SRAM initialization at the end of power on self-test.<br>0b - Disable SRAM initialization.<br>1b - Enable SRAM initialization. (recommended) |
| **4-0**<br>**POR_ALG_SEL** | RW | MBIST algorithm select for power on self-test.<br>11001b - Basic test. (recommended)<br>11111b - Full test. |

### 40.6.13 STCU_MBIST_DONE_STATUS (MBIST Done Status Register)

Offset : 3Ch

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

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | HSM_ROM | SYS_ROM | DMA_FLEXCAN | HSM | SUBSYS | CACHE_CPU0 | DTCM1_CPU0 | DTCM0_CPU0 | ITCM_CPU0 | SRAM |
| W | | | | | | | w0c | w0c | w0c | w0c | w0c | w0c | w0c | w0c | w0c | w0c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_MBIST_DONE_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **9**<br>**HSM_ROM** | W0C | HSM ROM MBIST Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |
| **8**<br>**SYS_ROM** | W0C | System ROM MBIST Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |
| **7**<br>**DMA_FLEXCAN** | W0C | DMA_FLEXCAN SRAM MBIST Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |
| **6**<br>**HSM** | W0C | HSM SRAM MBIST Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |
| **5**<br>**SUBSYS** | W0C | HSM/FlexCore Subsystem SRAM MBIST Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |
| **4**<br>**CACHE_CPU0** | W0C | CACHE_CPU0 SRAM MBIST Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |
| **3**<br>**DTCM1_CPU0** | W0C | DTCM1_CPU0 SRAM MBIST Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |
| **2**<br>**DTCM0_CPU0** | W0C | DTCM0_CPU0 SRAM MBIST Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |
| **1**<br>**ITCM_CPU0** | W0C | ITCM_CPU0 SRAM MBIST Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |

| | | |
|---|---|---|
| **0**<br>**SRAM** | W0C | System SRAM MBIST Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |

### 40.6.14 STCU_MBIST_FAIL_STATUS (MBIST Fail Status Register)

Offset : 40h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | HSM_ROM | SYS_ROM | DMA_FLEXCAN | HSM | SUBSYS | CACHE_CPU0 | DTCM1_CPU0 | DTCM0_CPU0 | ITCM_CPU0 | SRAM |
| W | | | | | | | w0c | w0c | w0c | w0c | w0c | w0c | w0c | w0c | w0c | w0c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_MBIST_FAIL_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **9**<br>**HSM_ROM** | W0C | HSM ROM MBIST Fail Flag<br>0b - No error occurs.<br>1b - Error occurs. |
| **8**<br>**SYS_ROM** | W0C | System ROM MBIST Fail Flag<br>0b - No error occurs.<br>1b - Error occurs. |
| **7**<br>**DMA_FLEXCAN** | W0C | DMA_FLEXCAN SRAM MBIST Fail Flag<br>0b - No error occurs.<br>1b - Error occurs. |
| **6**<br>**HSM** | W0C | HSM SRAM MBIST Fail Flag<br>0b - No error occurs.<br>1b - Error occurs. |
| **5**<br>**SUBSYS** | W0C | HSM/FlexCore Subsystem SRAM MBIST Fail Flag<br>0b - No error occurs.<br>1b - Error occurs. |

| | | |
|---|---|---|
| **4**<br>**CACHE_CPU0** | W0C | CACHE_CPU0 SRAM MBIST Fail Flag<br>0b - No error occurs.<br>1b - Error occurs. |
| **3**<br>**DTCM1_CPU0** | W0C | DTCM1_CPU0 SRAM MBIST Fail Flag<br>0b - No error occurs.<br>1b - Error occurs. |
| **2**<br>**DTCM0_CPU0** | W0C | DTCM0_CPU0 SRAM MBIST Fail Flag<br>0b - No error occurs.<br>1b - Error occurs. |
| **1**<br>**ITCM_CPU0** | W0C | ITCM_CPU0 SRAM MBIST Fail Flag<br>0b - No error occurs.<br>1b - Error occurs. |
| **0**<br>**SRAM** | W0C | System SRAM MBIST Fail Flag<br>0b - No error occurs.<br>1b - Error occurs. |

### 40.6.15 STCU_USER_GP (User General Purpose Register)

Offset : 44h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | UG | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | UG | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_USER_GP field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**UG** | RW | General Purpose Register<br>For users to store information, only reset by POR. |

### 40.6.16 STCU_SRAM_INI_CTRL (SRAM Initialization Control Register)

Offset : 48h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | MODE | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | LOCK | EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_SRAM_INI_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **17-16**<br>**MODE** | RW | Hardware SRAM initialization mode for different scenarios.<br>The bits should be configured to the corresponding value when trigger initialization after wakeup from different standby modes to avoid the retained SRAM blocks being initialized.<br>00b - After POR or wakeup from STANDBY0.<br>01b - After wakeup from STANDBY1.<br>10b - After wakeup from STANDBY2.<br>11b - After wakeup from STANDBY3. |
| **1**<br>**LOCK** | RW | Lock Control for Function Safety<br>Writing 1 will lock SRAM initialization enable. Only can be cleared by system reset. |
| **0**<br>**EN** | RW | Hardware SRAM Initialization Enable<br>Writing 1 will trigger initialization. During initialization, writing 0 will abort initialization. |

### 40.6.17 STCU_SRAM_INI_STATUS (SRAM Initialization Status Register)

Offset : 4Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | ABORT | BUSY | DONE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_SRAM_INI_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **2**<br>**ABORT** | RO | Hardware SRAM Initialization Abort Flag<br>0b - No abort event occurs.<br>1b - Abort event occurs. |
| **1**<br>**BUSY** | RO | Hardware SRAM Initialization Busy Flag<br>0b - No busy event occurs.<br>1b - Busy event occurs. |
| **0**<br>**DONE** | RO | Hardware SRAM Initialization Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |

### 40.6.18 STCU_SRAM_INI_SEL (SRAM Initialization Select Register)

Offset : 50h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | DTCM1_CPU0 | DTCM0_CPU0 | ITCM_CPU0 | SRAM |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_SRAM_INI_SEL field descriptions**

| Field | Type | Description |
|---|---|---|
| **3**<br>**DTCM1_CPU0** | RW | DTCM1_CPU0 SRAM Select for Hardware SRAM Initialization<br>0b - Not select.<br>1b - Select. |

| | | |
|---|---|---|
| **2**<br>**DTCM0_CPU0** | RW | DTCM0_CPU0 SRAM Select for Hardware SRAM Initialization<br>0b - Not select.<br>1b - Select. |
| **1**<br>**ITCM_CPU0** | RW | ITCM_CPU0 SRAM Select for Hardware SRAM Initialization<br>0b - Not select.<br>1b - Select. |
| **0**<br>**SRAM** | RW | System SRAM Select for Hardware SRAM Initialization<br>0b - Not select.<br>1b - Select. |

### 40.6.19 STCU_SRAM_INI_DONE_STATUS (SRAM Initialization Done Status Register)

Offset : 54h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | DTCM1_CPU0 | DTCM0_CPU0 | ITCM_CPU0 | SRAM |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_SRAM_INI_DONE_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **3**<br>**DTCM1_CPU0** | RW | DTCM1_CPU0 SRAM Initialization Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |
| **2**<br>**DTCM0_CPU0** | RW | DTCM0_CPU0 SRAM Initialization Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |
| **1**<br>**ITCM_CPU0** | RW | ITCM_CPU0 SRAM Initialization Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |

| | | |
|---|---|---|
| **0**<br>**SRAM** | RW | System SRAM Initialization Done Flag<br>0b - No done event occurs.<br>1b - Done event occurs. |

### 40.6.20 STCU_IRQ (Interrupt Request Register)

Offset : 58h

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

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | SEQ_ERR | SIZE_ERR |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**STCU_IRQ field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**EN** | RW | STCU Interrupt Enable<br>0b - Disable interruption.<br>1b - Enable interruption. |
| **1**<br>**SEQ_ERR** | RW | Illegal Register Access Sequence IRQ Flag<br>0b - No error occurs.<br>1b - Error occurs. |
| **0**<br>**SIZE_ERR** | RW | STCU Register Access Size Error IRQ Flag<br>0b - No error occurs.<br>1b - Error occurs. |

# Chapter 41 Cyclic Redundancy Check (CRC)

## 41.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 41.2 Chip-specific Information

N/A

## 41.3 Introduction

The Cyclic Redundancy Check (CRC) module generates 8/16/32-bit CRC code for error detection that may occur after data transmission or storage. The user can select 16/32-bit code to be calculated for 32 bits of data at a time. At the same time, a programmable polynomial and some parameters are necessary to implement the 8/16/32-bit CRC standard for the CRC module.

## 41.4 Features

The CRC module includes the following features:

- Programmable initial seed value and polynomial
- Support the bit/byte swap operation on input/output data
- Option for complement of final CRC result
- 32-bit CPU register programming interface
- Hardware CRC generator circuit using a 16-bit or 32-bit programmable shift register

## 41.5 Functional Description

### 41.5.1 CRC Initialization/Reinitialization

The user must set CRC_CR[WAS], program the CRC_POLY and some necessary parameters before the CRC calculation. When asserting CRC_CR[WAS], it shows that write to the CRC data register is the seed value.

After the CRC calculation is completed, the module can be reinitialized for a new CRC computation. The user can reassert CRC_CR[WAS] and program a new, or previously used, seed value. The user must set all other parameters before programming the seed value and subsequent data values.

### 41.5.2 CRC Calculations

To get a correct CRC computation, all the data should be contiguous in 16-bit or 32-bit CRC modes. But data values can be programmed into different bits at a time, the user can select 8 bits, 16 bits, or 32 bits at a time.

### 41.5.3 8-bit CRC/16-bit CRC/32-bit CRC

The user can compute the 8-bit/16-bit/32-bit CRC by performing the following steps:

1) Set CRC_CR[TCRC8] to enable 8-bit CRC mode. Keep CRC_CR[TCRC8] to 0 while set CRC_CR[TCRC] to enable 16-bit/32-bit CRC mode.
2) Program the data swap and complement options in the CTRL register as required for the CRC calculation.
3) Write a polynomial to the CRC_POLY field, for 16-bit CRC mode, the user should write a 16-bit polynomial and the CRC[HIGH] is unable.
4) Set CRC_CR[WAS] to program the seed value.
5) Write seed data to CRC_DATA, for 16-bit CRC mode, CRC_DATA[HU:HL] are not used.
6) Clear the CRC_CR[WAS] to start writing data values.
7) Write data values into CRC_DATA[HU:HL:LU:LL]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA, for 16-bit CRC mode, the CRC results will only be stored into CRC_DATA[LU:LL].
8) When all values have been written, read the final CRC result from CRC_DATA, for 16-bit mode, read the final CRC result from CRC_DATA[LU:LL].

### 41.5.4 Data Swap Feature

The CRC module is designed to support data swap for some CRC standards. These CRC standards require the input data and/or the final checksum to be swapped. The user can set each data swap individually by configuring the different options and data is swapped on the fly on being read or written.

### 41.5.5 Types of Data Swap

The CRC module supports four types of swap functions to flip the bits and bytes. The user can set CR[DSW] or CR[DSR] for both writing input data and reading the CRC result. The swap condition is shown in the table below:

| CR[DSW] or CR[DSR] | Description | Result |
|---|---|---|
| 00 | No swap occurs. | reg [31:0] |
| 01 | Bits in a byte are swapped, while bytes are not swapped. | {reg [24:31], reg [16:23], reg [8:15], reg [ 0:7]} |
| 10 | Both bits in bytes and bytes are swapped. | {reg [0:7], reg [8:15], reg [16:23], reg [24:31]} |
| 11 | Bytes are swap, but bits are not swapped | {reg [7:0], reg [15:8], reg [23:16], reg [31:24]} |

In addition, the CRC is calculated on the valid byte(s) only. For 8-bit and 16-bit write accesses to the CRC data register, the data will swap with zeros to complete the unused bits or bytes.

### 41.5.6 CRC Result Complement

The CRC[FOXR] is used to complement the checksum. When the user reads the CRC data registers, the complement function will work on the data value and store the data into the CRC data register. If the CR[FXOR] is cleared, the user can get the raw checksum value by reading the CRC data register.

## 41.6 Register Definition

CRC base address : 4002_A000h

**Register Overview - CRC**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | CRC_DATA (Data Register) | 32 | RW | FFFF_FFFFh | page |
| **4h** | CRC_POLY (Polynomial Register) | 32 | RW | 0000_1021h | page |
| **8h** | CRC_CR (Control Register) | 32 | RW | 0000_0000h | page |
| **Ch** | CRC_SDAT (Data Bit Swap Register) | 32 | RW | 0000_0000h | page |

### 41.6.1 CRC_DATA (Data Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HU | | | | | | | | HL | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LU | | | | | | | | LL | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**CRC_DATA field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**HU** | RW | CRC HIGH Upper Byte<br>This byte is unable for 16-bit CRC mode to program a seed value, but this field will store the data for the CRC checksum generation in both 16-bit and 32-bit CRC modes |
| **23-16**<br>**HL** | RW | CRC HIGH Lower Byte<br>This byte is unable for 16-bit CRC mode to program a seed value, but this field will store the data for the CRC checksum generation in both 16-bit and 32-bit CRC modes |
| **15-8**<br>**LU** | RW | CRC Low Upper Byte<br>When CR[WAS] is 1, this field will be used to store part of the seed value. Otherwise<br>data written to this field will be used for CRC checksum generation. |
| **7-0**<br>**LL** | RW | CRC Low Lower Byte<br>When CR[WAS] is 1, this field will be used to store part of the seed value. Otherwise<br>data written to this field will be used for CRC checksum generation. |

### 41.6.2 CRC_POLY (Polynomial Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HIGH | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LOW | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

**CRC_POLY field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-16<br>HIGH** | RW | Writable and readable in 32-bit CRC mode (CR[TCRC] is 1). This field is not writable in 16-bit CRC mode (CR[TCRC] is 0). |
| **15-0<br>LOW** | RW | Writable and readable in both 32-bit and 16-bit CRC modes. |

### 41.6.3 CRC_CR (Control Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DSW | | DSR | | TCRC | FXOR | WAS | TCRC | 0 | | | | | | | |
| W | | | | | 8 | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CRC_CR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-30<br>DSW** | RW | Data Swap for Writes<br>Defines the data swap configuration of the data written to the CRC data register. See the description of the data swap feature for the available swap options.<br>00b - No data swap.<br>01b - Bits in bytes are swap; bytes are not swap.<br>10b - Both bits in bytes and bytes are swap.<br>11b - Only bytes are swap; no bits in a byte are swap. |
| **29-28<br>DSR** | RW | Data Swap for Read<br>Identifies the data swap configuration of the value read from the CRC Data register. See the description of the data swap feature for the available swap options.<br>00b - No data swap.<br>01b - Bits in bytes are swap; bytes are not swap.<br>10b - Both bits in bytes and bytes are swap.<br>11b - Only bytes are swap; no bits in a byte are swap. |

| | | |
|---|---|---|
| **27**<br>**TCRC8** | RW | Enable 8 Bits Data CRC Calculation<br>Prior than TCRC bit. This bit should set to 0b when 16/32-bits CRC protocol is configured.<br>0b - Disable 8-bits CRC protocol.<br>1b - Enable 8-bits CRC protocol. |
| **26**<br>**FXOR** | RW | Complement Read Of CRC Data Register<br>Some CRC protocols require the final checksum to be XORed with 0xFFFFFFFF or 0xFFFF. Asserting<br>this bit enables on the fly complementing of read data.<br>0b - No XOR on reading.<br>1b - Invert or complement the read value of the CRC Data register. |
| **25**<br>**WAS** | RW | Write CRC Data Register As Seed<br>0b - Writes to the CRC data register are data values.<br>1b - Writes to the CRC data register are seed values. |
| **24**<br>**TCRC** | RW | Width of CRC Protocol<br>0b - 16-bit CRC protocol.<br>1b - 32-bit CRC protocol. |

### 41.6.4 CRC_SDAT (Data Bit Swap Register)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | HIGH | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LOW | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CRC_SDAT field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-16**<br>**HIGH** | RO | Bit Swap Result Upper 2 Bytes<br>Bit swap results are always stored in highest bytes. |
| **15-0**<br>**LOW** | RO | Bit Swap Result Lower 2 Bytes |

# Chapter 42 General-Purpose Input/Output (GPIO)

## 42.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 42.2 Features

The General-Purpose Input/Output (GPIO) module contains the following features:

- Port Data Input register.
- Port Data Output register with corresponding set/clear/toggle registers.
- Port Data Direction register.

## 42.3 Overview

When a pin is configured as the GPIO function, the GPIO data direction and output data registers control the direction and output data. When the pin is configured as any digital function, the GPIO input data register shows the logical value on each pin, provided that the corresponding Port Control (PORT) module of that pin input is enabled.

## 42.4 Functional Description

### 42.4.1 Domain Write Protection (DWP)

The GPIO has the domain write protection. Each pin of GPIO is controlled by the DWP of the corresponding PORT_PCR. See more details in Section *Domain Write Protection* in Chapter *PORT*.

### 42.4.2 General-Purpose Input

The Port Data Input register (PDIR) can obtain the logical status of each pin, provided that the corresponding Port Control (PORT) module is enabled, the pin is configured as a digital function, and the corresponding bit in the Port Input Disable register (PIDR) is cleared.

### 42.4.3 General-Purpose Output

Port Data Output Register (PDOR) and Port Data Direction Register (PDDR) control the logical state of each pin, provided that the pin is configured as the GPIO function.

There are pin data set, pin data clear and pin data toggle registers to allow outputs of one or more pins to be set, cleared, or toggled for a port.

Flagchip

## 42.5 Register Definition

GPIOA base address : 4100_0000h

GPIOB base address : 4100_0040h

GPIOC base address : 4100_0080h

GPIOD base address : 4100_00C0h

GPIOE base address : 4100_0100h

**Register Overview - GPIO**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | GPIO_PDOR (Port Data Output Register) | 32 | RW | 0000_0000h | page |
| **4h** | GPIO_PSOR (Port Set Output Register) | 32 | WO | 0000_0000h | page |
| **8h** | GPIO_PCOR (Port Clear Output Register) | 32 | WO | 0000_0000h | page |
| **Ch** | GPIO_PTOR (Port Toggle Output Register) | 32 | WO | 0000_0000h | page |
| **10h** | GPIO_PDIR (Port Data Input Register) | 32 | RO | 0000_0000h | page |
| **14h** | GPIO_PDDR (Port Data Direction Register) | 32 | RW | 0000_0000h | page |
| **18h** | GPIO_PIDR (Port Input Disable Register) | 32 | RW | 0000_0000h | page |

### 42.5.1 GPIO_PDOR (Port Data Output Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDO | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDO | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**GPIO_PDOR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-0**<br>**PDO** | RW | Port Data Output<br>0b - If the pin is configured as a general-purpose output, drive logic 0 to pin.<br>1b - If the pin is configured as a general-purpose output, drive logic 1 to pin. |

### 42.5.2 GPIO_PSOR (Port Set Output Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PSO | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PSO | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**GPIO_PSOR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**PSO** | WO | Port Set Output<br>Writing this register will update bits in the PDOR accordingly, as shown below:<br>0b - No effect.<br>1b - Set the corresponding bit to logic 1. |

### 42.5.3 GPIO_PCOR (Port Clear Output Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PCO | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | PCO | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**GPIO_PCOR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**PCO** | WO | Port Clear Output<br>Writing this register will update bits in the PDOR accordingly, as shown below:<br>0b - No effect.<br>1b - Clear the corresponding bit to logic 0. |

### 42.5.4 GPIO_PTOR (Port Toggle Output Register)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PTO | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | PTO | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**GPIO_PTOR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**PTO** | WO | Port Toggle Output<br>Writing this register will update bits in the PDOR accordingly, as shown below:<br>0b - No effect.<br>1b - Inverse the corresponding bit. |

### 42.5.5 GPIO_PDIR (Port Data Input Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PDI | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PDI | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**GPIO_PDIR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**PDI** | RO | Port Data Input<br>If the pin is not configured as a digital function, the corresponding bit reads 0.<br>0b - Pin value is logic 0 or pin is not configured as a digital function.<br>1b - Pin value is logic 1. |

### 42.5.6 GPIO_PDDR (Port Data Direction Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PDD | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PDD | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**GPIO_PDDR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**PDD** | RW | Port Data Direction<br>Configure the corresponding port pins as an input or output.<br>0b - Configured as input.<br>1b - Configured as output. |

### 42.5.7 GPIO_PIDR (Port Input Disable Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PID | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PID | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**GPIO_PIDR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**PID** | RW | Port Input Disable<br>0b - Configured as input is allowed.<br>1b - Configured as input is not allowed. The corresponding PDIR bit reads logic 0. |

# Chapter 43 Port Control (PORT)

## 43.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 43.2 Chip-specific Information

N/A

## 43.3 Features

The Port Control (PORT) module contains the following features:

- Port control
  - Pull-enable (pull-up/pull-down) and pull-disable function.
  - Slew rate control.
  - Input passive filter field.
  - Drive Strength Control.
  - Up to 8 functions can be selected by mux control.
  - Emergency stop when the FCSMU detects a system error and generates a fault.
- Pin interrupt
  - Interrupt enable and flag.
  - Edge sensitive (rising, falling, or both) or level sensitive (low or high) configuration.
  - Can be configured as an interrupt or a DMA request.
  - Asynchronous wakeup in low-power modes; can be configured to asynchronous wake-up without interruption.
  - Can be configured as an asynchronous wakeup source only without generating an interrupt or a DMA request.
- Digital input filter
  - Enable or bypass the control field.
  - Optional clock source.
  - Filter size has a five-bit resolution.
- Domain Write Protection (DWP)
  - Support write access control for three masters, each of which is controlled individually.
  - Global registers and pin registers are controlled by different DWP fields.
  - Write permission of the GPIO is controlled by the DWP field which controls the corresponding pin register.

## 43.4 Overview

Pins may not be configured with all functions in the PORT_PCRn register. Not all pins of each port are used.

Padring power domain is divided into PD0 and PD1. Only PD0 pads can work under standby mode. A high-speed pad is added to the PD1. PCRn[DSE] is used for the high-speed pad and PCRn[SRE] is used for the normal pad. Refer to the IO description in the appendix for details.

***Note:***

1) If PTA11 is configured as NMI_b, the user needs to clear the SCM_INT_ROUTER_NMI register, configure PTA11 as NMI_b, and then configure the SCM_INT_ROUTER_NMI as the previous value.
2) To use the digital filter function, the function should be configured first before configuring other functions.

## 43.5 Functional Description

### 43.5.1 Domain Write Protection

PORT has two kinds of domain write protection: global DWP (GLDWP[DWP]) and pin DWP (PCRn[DWP]). The following table is the control range for both DWPs.

| Type | Control Range |
|---|---|
| **Global DWP** | GPCLR, GPCHR, GICLR, GICHR, DFER, DFCR, DFWR |
| **Pin DWP** | PCRn, GPIO related |

If DWPLK is set, the corresponding DWP bit cannot be written. If both DWP and DWPLK are configured at the same time, the DWP configuration is invalid. The correct order is to configure DWP before configuring DWPLK.

The DWP field can control the write permissions of the following masters: CPU0, DMA0, and FlexCore.

### 43.5.2 Pin Control

If a pin is used as a digital function, the input buffer of the pin will be enabled, allowing the pin state to be read by the corresponding GPIO Port Data Input register (GPIO_PDIR), and a pin interrupt or a DMA request can be generated.

An input pin may float if it is not connected to a driver. A floating pin should be avoided when its input buffer is enabled, as this may result in increased power consumption. To ensure that a pin does not float when its input buffer is enabled, internal pull resistance can be enabled (or use an external pull resistor). If the output buffer is enabled, the internal pull resistor will be automatically disabled, allowing the Pull Enable bit to remain set without causing leakage. If the pin is not used, multiplexing configuration is set to disabled or analog function, which can reduce power consumption.

### 43.5.3 Global Pin Control

The purpose of the two global pin control registers is to configure multiple pins in one register to achieve the same peripheral functions. Each global pin control register can control the lower half of up to the 16 control registers, but cannot write to a locked register.

### 43.5.4 Global Interrupt Control

The purpose of the two global interrupt control registers is to configure multiple pins in one register to achieve the same peripheral functions. Each global interrupt control register can control the upper half of up to the 16 control registers.

### 43.5.5 External Interrupts

The input level or edge change can generate a pin interrupt, and an interrupt can also be generated even in low power mode.

### 43.5.6 Digital Filter

Digital filter clock source and width configuration are shared by all digital filters in one port, and must be changed only if all digital filters in that port are disabled.

A digital filter clock can be selected between the bus clock and the AON32K_CLK clock. If digital filters of a port are configured to use the bus clock, the digital filters will be bypassed during stop mode.

When a digital filter is enabled, the input is synchronized with the filter clock. The value of the filter input is updated to the filter output only if the synchronized input and the output of the digital filter remain different for the number of filter clock cycles equal to the value configured in the filter width register.

When calculating the maximum latency of the digital filter, three additional filter clock cycles need to be added to the filter width configuration register.

### 43.5.7 Safety Related

If a pin enables the emergency stop, once the FCSMU detects a system error and generates a fault, the pin output function will be disabled to avoid sending out wrong data.

## 43.6 Register Definition

PORTA base address : 4004_7000h

PORTB base address : 4004_8000h

PORTC base address : 4004_9000h

PORTD base address : 4004_A000h

PORTE base address : 4004_B000h

**Register Overview - PORT**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h+n*4h** | PORT_PCRn (Port Control Register) | 32 | RW | 0000_0000h | page |
| **80h** | PORT_GPCLR (Global Pin Control Low Register) | 32 | WO | 0000_0000h | page |
| **84h** | PORT_GPCHR (Global Pin Control High Register) | 32 | WO | 0000_0000h | page |
| **88h** | PORT_GICLR (Global Interrupt Control Low Register) | 32 | WO | 0000_0000h | page |
| **8Ch** | PORT_GICHR (Global Interrupt Control High Register) | 32 | WO | 0000_0000h | page |
| **A0h** | PORT_ISFR (Interrupt Status Flag Register) | 32 | W1C | 0000_0000h | page |
| **C0h** | PORT_DFER (Digital Filter Enable Register) | 32 | RW | 0000_0000h | page |
| **C4h** | PORT_DFCR (Digital Filter Clock Register) | 32 | RW | 0000_0000h | page |
| **C8h** | PORT_DFWR (Digital Filter Width Register) | 32 | RW | 0000_0000h | page |
| **CCh** | PORT_GLDWP (Global Domain Write Protection Register) | 32 | RW | 0000_0000h | page |

### 43.6.1 PORT_PCRn (Port Control Register)

Offset : 0h+N*4h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DWPLK | 0 | DWP | | | | | ISF | 0 | | ESTOP | WKUDIS | IRQC | | | |
| W | | | | | | | | w1c | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | MUX | | | DSE1 | DSE0 | ODE | PFE | 0 | SRE | PE | PS |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORT_PCRn field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31**<br>**DWPLK** | RW | Domain Write Protection Lock<br>0b - DWP can be written<br>1b - DWP cannot be written |
| **29-25**<br>**DWP** | RW | Domain Write Protection<br>This field indicates which core or DMA is allowed to write this pin register and GPIO register corresponding to this pin.<br>Bit 29 - FlexCore<br>Bit 28 - DMA0<br>Bit 27 - Reserved<br>Bit 26 - Reserved<br>Bit 25 - CPU0<br>For each bit:<br>0b - Allowed to write.<br>1b - Not allowed to write. |
| **24**<br>**ISF** | W1C | Interrupt Status Flag<br>Pin interrupts can be configured in all digital pin multiplexing modes.<br>0b - No interrupt detected.<br>1b - Interrupt detected.<br>Note:<br>If an interrupt is configured as a DMA request, the flag will be cleared automatically when the DMA transfer is completed. Otherwise, this flag bit needs to be cleared by writing a logic 1 to it.<br>If interrupt configuration is level sensitive and the level is valid, this flag bit will still be set again after being cleared. |
| **21**<br>**ESTOP** | RW | Emergency Stop<br>Emergency Stop is valid if the pin is configured as a digital function.<br>If the FCSMU generates a fault and pin emergency stop is enabled, pin output function will be disabled.<br>0b - Emergency stop function is disabled.<br>1b - Emergency stop function is enabled. |
| **20**<br>**WKUDIS** | RW | Wakeup Disable<br>This field controls if the pin will only be used for wakeup.<br>1b - Pin interrupt function only enabled during low power mode (standby & stop mode, wait mode is not included).<br>0b - Pin interrupt function enabled under all power modes |

| | | |
|---|---|---|
| **19-16**<br>**IRQC** | RW | Interrupt Configuration<br>Pin interrupts can be configured in all digital pin multiplexing modes.<br>Valid configuration is as follows:<br>0000b - Disabled.<br>0001b - DMA request and ISF flag on the rising edge.<br>0010b - DMA request and ISF flag on the falling edge.<br>0011b - DMA request and ISF flag on either edge.<br>0100b - Reserved.<br>0101b - Reserved.<br>0110b - Reserved.<br>0111b - Reserved.<br>1000b - Interrupt and ISF flag when logic 0.<br>1001b - Interrupt and ISF flag on the rising edge.<br>1010b - Interrupt and ISF flag on the falling edge.<br>1011b - Interrupt and ISF flag on either edge.<br>1100b - Interrupt and ISF flag when logic 1.<br>1101b - TRGSEL input, signal is consistent with PAD input.<br>1110b - TRGSEL input, signal is opposite with PAD input.<br>1111b - Reserved. |
| **15**<br>**LK** | RW | Lock Control<br>0b - PCR bits [15:0] are not locked.<br>1b - PCR bits [15:0] are locked and cannot be updated until the next system reset. |
| **10-8**<br>**MUX** | RW | Mux Configuration<br>Not all MUX can be selected for each pin. See Pinout in the appendix for specific configuration:<br>000b - ALT 0 (specific analog function).<br>001b - ALT 1 (GPIO).<br>010b - ALT 2 (specific digital function).<br>011b - ALT 3 (specific digital function).<br>100b - ALT 4 (specific digital function).<br>101b - ALT 5 (specific digital function).<br>110b - ALT 6 (specific digital function).<br>111b - ALT 7 (specific digital function). |
| **7**<br>**DSE1** | RW | Drive Strength 1 Enable<br>Drive strength is valid if the pin is configured as a digital output.<br>Note: Only valid for UHS pads. DSE1 should not be enabled before enabling DSE0.<br>0b - Low drive strength.<br>1b - High drive strength. |

| | | |
|---|---|---|
| **6**<br>**DSE0** | RW | Drive Strength 0 Enable<br>Drive strength is valid if the pin is configured as a digital output.<br>Note: Only valid for HS pads.<br>0b - Low drive strength.<br>1b - High drive strength. |
| **5**<br>**ODE** | RW | Open Drain Enable<br>Open drain is valid if the pin is configured as a digital function.<br>0b - Open drain is disabled.<br>1b - Open drain is enabled. |
| **4**<br>**PFE** | RW | Passive Filter Enable<br>Passive Filter is valid if the pin is configured as a digital input. Specially, this bit is only valid for pins which have the digital function NMI_b or RESET.<br>0b - Passive filter is disabled.<br>1b - Passive filter is enabled. |
| **2**<br>**SRE** | RW | Slew Rate Enable<br>Slew Rate is valid if the pin is configured as a digital function.<br>Note: Only valid for normal pads.<br>0b - Fastest slew rate<br>1b - Slowest slew rate |
| **1**<br>**PE** | RW | Pull Enable<br>Pull Enable is valid if the pin is configured as a digital input.<br>0b - Internal pullup or pulldown resistor is not enabled.<br>1b - Internal pullup or pulldown resistor is enabled. |
| **0**<br>**PS** | RW | Pull Select<br>Pull Select is valid if the pin is configured as a digital input and PE is set.<br>0b - Internal pulldown resistor is enabled.<br>1b - Internal pullup resistor is enabled. |

### 43.6.2 PORT_GPCLR (Global Pin Control Low Register)

Offset : 80h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | | | | | | |
| W | GPWE | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | GPWD | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORT_GPCLR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-16**<br>**GPWE** | WO | Global Pin Write Enable<br>Update the value of GPWD to bits [15:0] of PCR0 - 15. The update will be ignored if the corresponding PCR register is locked.<br>0b - Corresponding PCR Register won't be updated.<br>1b - Corresponding PCR Register will be updated. |
| **15-0**<br>**GPWD** | WO | Global Pin Write Data<br>Value that is written to PCR register bits [15:0] selected by GPWE. |

### 43.6.3 PORT_GPCHR (Global Pin Control High Register)

Offset : 84h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | | | | | | |
| W | GPWE | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GPWD | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORT_GPCHR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-16**<br>**GPWE** | WO | Global Pin Write Enable<br>Update the value of GPWD to bits [15:0] of PCR16 - 31. The update will be ignored if the corresponding PCR register is locked.<br>0b - Corresponding PCR Register won't be updated.<br>1b - Corresponding PCR Register will be updated. |
| **15-0**<br>**GPWD** | WO | Global Pin Write Data<br>Value that is written to PCR register bits [15:0] selected by GPWE. |

### 43.6.4 PORT_GICLR (Global Interrupt Control Low Register)

Offset : 88h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | | | | | | |
| W | GIWD | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWE | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORT_GICLR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-16**<br>**GIWD** | WO | Global Interrupt Write Data<br>Value that is written to PCR register bits [31:16] selected by GPWE. |
| **15-0**<br>**GIWE** | WO | Global Interrupt Write Enable<br>Update the value of GIWD to bits [31:16] of PCR0 - 15.<br>0b - Corresponding PCR Register won't be updated.<br>1b - Corresponding PCR Register will be updated. |

### 43.6.5 PORT_GICHR (Global Interrupt Control High Register)

Offset : 8Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | | | | | | |
| W | GIWD | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | GIWE | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORT_GICHR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-16**<br>**GIWD** | WO | Global Interrupt Write Data<br>Value that is written to PCR register bits [31:16] selected by GIWE. |
| **15-0**<br>**GIWE** | WO | Global Interrupt Write Enable<br>Update the value of GIWD to bits [31:16] of PCR16 - 31.<br>0b - Corresponding PCR Register won't be updated.<br>1b - Corresponding PCR Register will be updated. |

### 43.6.6 PORT_ISFR (Interrupt Status Flag Register)

Offset : A0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ISF | | | | | | | | | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ISF | | | | | | | | | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORT_ISFR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**ISF** | W1C | Interrupt Status Flag<br>Each bit in the field represents ISF of the corresponding PCR register.<br>0b - No interrupt detected.<br>1b - Interrupt detected. |

### 43.6.7 PORT_DFER (Digital Filter Enable Register)

Digital Filter is for GPIO only. Do not use it if the pin uses other ALT options.

Offset : C0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DFE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DFE | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORT_DFER field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**DFE** | RW | Digital Filter Enable<br>Digital Filter configuration is valid if the pin is configured as a digital input. Each bit controls DFE of the corresponding pin.<br>0b - Digital filter is disabled.<br>1b - Digital filter is enabled. |

### 43.6.8 PORT_DFCR (Digital Filter Clock Register)

Digital Filter is for GPIO only. Do not use it if the pin uses other ALT options.

Offset : C4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | | | | | | | | | | | | | | | CS |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORT_DFCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**CS** | RW | Clock Source<br>Digital Filter configuration is valid if the pin is configured as a digital input. The filter clock source can be changed only if all digital filters are disabled.<br>0b - Select the bus clock as the clock source.<br>1b - Select the AON32K_CLK clock as the clock source. |

### 43.6.9 PORT_DFWR (Digital Filter Width Register)

Digital Filter is for GPIO only. Do not use it if the pin uses other ALT options.

Offset : C8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | FILT | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORT_DFWR field descriptions**

| Field | Type | Description |
|---|---|---|
| **4-0**<br>**FILT** | RW | Filter Length<br>Digital Filter configuration is valid if the pin is configured as a digital input. Configure the maximum glitch size (in clock cycles) filtered by the digital filter when the digital filter is enabled. The filter length can be changed only if all digital filters are disabled. |

### 43.6.10 PORT_GLDWP (Global Domain Write Protection Register)

Offset : CCh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DWPLK | 0 | DWP | | | | | 0 | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PORT_GLDWP field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**DWPLK** | RW | Domain Write Protection Lock<br>0b - DWP can be written<br>1b - DWP cannot be written |
| **29-25**<br>**DWP** | RW | Domain Write Protection<br>This field indicates which core or DMA is allowed to write Global Registers<br>Bit 29 - FlexCore<br>Bit 28 - DMA0<br>Bit 27 - Reserved<br>Bit 26 - Reserved<br>Bit 25 - CPU0<br>For each bit:<br>0b - Allowed to write.<br>1b - Not allowed to write. |

Flagchip

# Chapter 44 Trigger Select (TRGSEL)

## 44.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 44.2 Chip-specific information

The FC7240F2M contains 6 Trigger Selects (TRGSELs). The detailed connection information is described in Chapter. Trigger Select (TRGSEL) Source.

## 44.3 Features

The TRGSEL module allows the software to select the trigger source for peripherals.

## 44.4 Overview

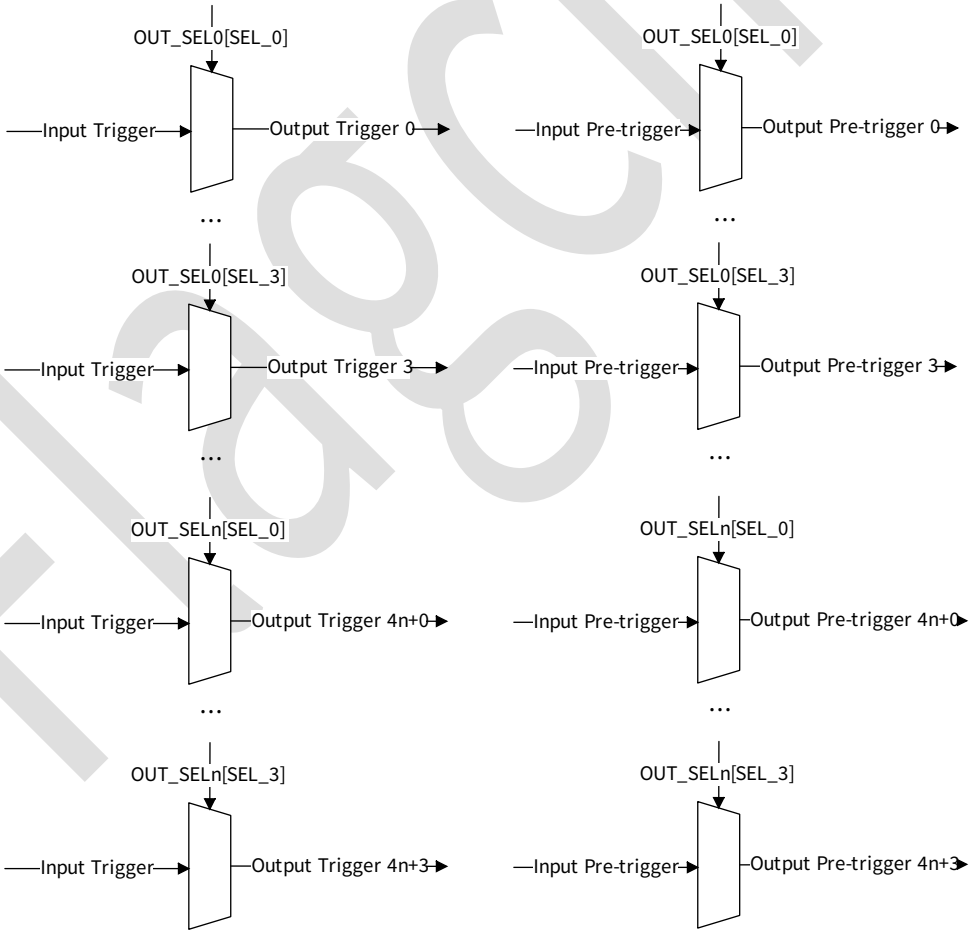
The TRGSEL provides an extremely flexible mechanism for connecting various trigger sources to multiple pins/peripherals.

With the TRGSEL, each control register supports up to four output triggers, and each output trigger can be selected from the available input triggers (e.g. No.1 - 127 in the following TRGSEL0 table). For example, to establish a path from "SCM0_SW_TRG0" (Input No.2) to "FTU7_TRG0" (Output No.7), write "2 (2'b10)" to TRGSEL0_OUT_SEL1[SEL_3] ("TRGSEL0_OUT_SEL1" is the register that supports "LU0_INPUT1D", and "SEL_3" is the bit that corresponds to Output No.7)

## 44.5 Block Diagram

The following is the block diagram of the TRGSEL module.

**Figure 69. TRGSEL block diagram**



## 44.6 Register Definition

TRGSEL0 base address : 4002_6000h

TRGSEL1 base address : 4002_7000h

TRGSEL2 base address : 4002_8000h

TRGSEL3 base address : 4002_9000h

TRGSEL4 base address : 4043_B000h

TRGSEL5 base address : 4043_C000h

**Register Overview - TRGSEL**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h+n*4h** | TRGSEL_OUT_SELn (OUT SEL for N) | 32 | RW | 0000_0000h | page |

### 44.6.1 TRGSEL_OUT_SELn (OUT SEL for N)

"N" means register slot. For example, offset 0 (32bit) controls outputs 0 - 3, and offset 4 controls outputs 4 - 7.

Note: Access to an un-supported register will generate a bus fault.

Offset : 0h+N*4h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LOCK | SEL_3 | | | | | | | 0 | SEL_2 | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | SEL_1 | | | | | | | 0 | SEL_0 | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TRGSEL_OUT_SELn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**LOCK** | RW | The LOCK bit can only be written once after any system reset. When asserted, writes to this register will be ignored until the next system reset clears LOCK. |
| **30-24**<br>**SEL_3** | RW | Select for output (N*4+3)<br>For example, if n = 1, this bit corresponds to Output No.7, and if n = 2, this bit corresponds to Output No.11. |

| | | |
|---|---|---|
| **22-16**<br>**SEL_2** | RW | Select for output (N*4+2)<br>For example, if n = 1, this bit corresponds to Output No.6, and if n = 2, this bit corresponds to Output No.10. |
| **14-8**<br>**SEL_1** | RW | Select for output (N*4+1)<br>For example, if n = 1, this bit corresponds to Output No.5, and if n = 2, this bit corresponds to Output No.9. |
| **6-0**<br>**SEL_0** | RW | Select for output (N*4+0)<br>For example, if n = 1, this bit corresponds to Output No.4, and if n = 2, this bit corresponds to Output No.8. |

Flagchip

# Chapter 45 Logic Unit (LU)

## 45.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 45.2 Chip-specific Information

For logic unit (LU) input (LU_INPUTnA, LU_INPUTnB, LU_INPUTnC and LU_INPUTnD) connection, refer to the TRGSEL0 table for details.

## 45.3 Features

The LU includes the following features:

- 4 Logic Generators (LGs); each LG contains 2 AND/OR/INVERT (AOI) and one configurable Flip-Flop (FF).
- Each AOI evaluates a combinational Boolean expression.
- One flexible FF can be configured as RS, D-FF, T-FF, JK-FF, or Latch.
- Each LG has four inputs (LU_INPUTA/B/C/D) and two outputs (LU_OUTA/B).
- The two outputs of LG can operate as hardware trigger signals or for other purposes.

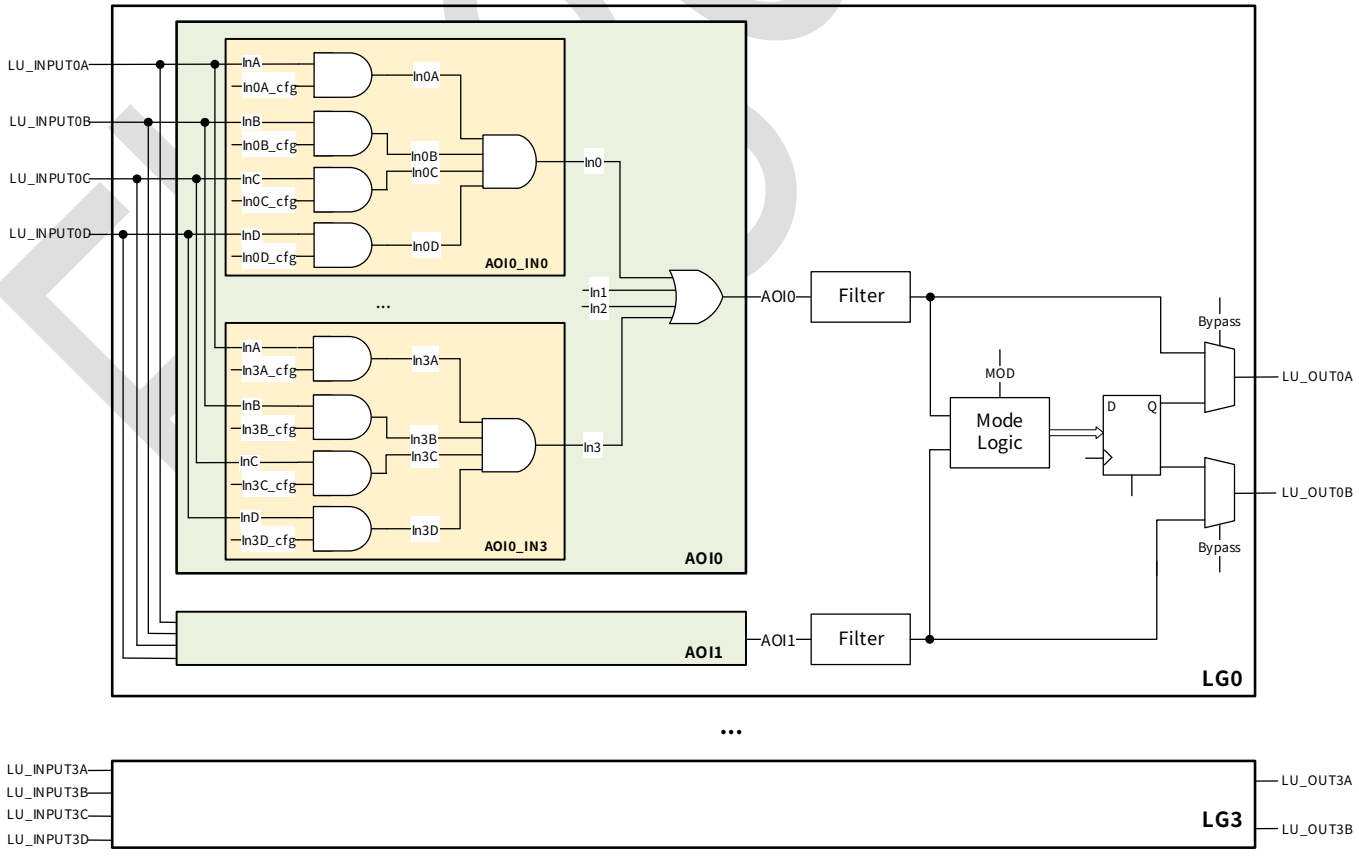
## 45.4 Overview

The LU module mainly includes two parts: Two AOI modules and one configurable FF. These two parts make the sub block called LG. There are 4 LGs in LU. It supports the generation of a configurable number of EVENT signals. The two AOI combinational expressions share the four associated LU inputs: LU_INPUTA, LU_INPUTB, LU_INPUTC, and LU_INPUTD. The FF can be configured to make the two expressions act as the RESET port, SET port or D port, CLK port or simply go through to LU output with FF bypassed.

## 45.5 Block Diagram

The following is the block diagram of the LU module.

**Figure 70. LU block diagram**



## 45.6 Functional Description

### 45.6.1 Input Mode Configuration

The LU supports the LU input sync and AOI output filter functions.

Both features are default disabled after reset. The input sync function is enabled by configuring CTRL[SYNC], and is implemented through 2 DFFs to ensure that no metastable conditions occur. Any glitch whose width is less than a bus clock period will be removed.

AOI output = In0 | In1 | In2 | In3, which In0 = In0A & In0B & In0C & In0D, In1 = In1A & In1B & In1C & In1D, etc. For In0A/B/C/D, is the output after sync logic from LU_INPUTA/B/C/D (configured by AOI register, for example, IN0A_CFG=00, means In0A = 0; IN0A_CFG=01, means In0A = LU_INPUTA; IN0A_CFG=10, means In0A = ~LU_INPUTA, and IN0A_CFG=11 indicate that In0A = 1).

The output filter function is enabled by configuring FILT[PER] to a non-zero value. This function uses the bus clock to filter the AOI output, and any input width less than "(FILT[CNT] + 3) x FILT[PER] + 2" will be filtered.

***Note:*** Due to the limitation of the digital filter, if the input is glitchy and the period toggles at the posedge of the clock, the output is not expected. And this is not in design consideration.

### 45.6.2 Output Mode Configuration

The LU module inside implements different kinds of flip-flops for the generation of desired LU output. A flip-flop can be configured as Bypass mode, RS trigger mode, T-FF mode, D-FF mode, JK-FF mode, or Latch mode.

#### *45.6.2.1 Bypass Mode*

When register bits CTRL[MOD] are configured as 0x0, Bypass mode will be selected. In this mode, the flip-flop will be bypassed, the two AOI expressions "AOI0" and "AOI1" will be directly assigned to LU outputs (LU_OUTA and LU_OUTB).

#### *45.6.2.2 RS Trigger Mode*

When register bits CTRL[MOD] are configured as 0x1, RS trigger mode will be selected.

In this mode, AOI0 is connected to the RS flip flop Reset, and AOI1 is connected to the RS flip flop Set. Both are active high. When "S" (Set) is high, whatever "R" (Reset) is, LU_OUTA will be "1," when "R" is high and "S" is low, LU_OUTA will be "0." If both "R" and "S" are low, LU output will be kept in the previous state.

LU_OUTB is always the complement of LU_OUTA.

#### *45.6.2.3 T-FF Mode*

When register bits CTRL[MOD] are configured as 0x2, T-FF mode will be selected.

In this mode, AOI0 is connected to T port of T-FF, AOI1 is connected to TFF CLK port. When T asserts, the Q port (LU_OUTA) will turn over at the rising edge of "CLK". When T dis-assert, Q(LU_OUTA) will be kept.

LU_OUTB is always the complement of LU_OUTA. In this mode, it is suggested to enable the input sync or filter function to remove the possible glitch.

#### *45.6.2.4 D-FF Mode*

When register bits CTRL[MOD] are configured as 0x3, D-FF mode will be selected.

In this mode, AOI0 is connected to D port of D-FF, AOI1 is connected to DFF CLK port. At the rising edge of "CLK", D will be captured to Q (LU_OUTA).

LU_OUTB is always the complement of LU_OUTA. In this mode, it is suggested to enable the input sync or filter function to remove the possible glitch.

#### *45.6.2.5 JK-FF Mode*

When register bits CTRL[MOD] are configured as 0x4, JK-FF mode will be selected.

In general, JK Flip-Flop has four input ports: J, K, Q, and CLK (Q is the output of Flip-Flop). And the logical expression is J&~Q | ~K&Q.

In this mode, reused as DFF mode, AOI0 is connected to D port, and AOI1 is connected to DFF CLK port, the FB_OVRD is configured to select 1 of 4 inputs replaced by output Q.

To achieve the JK-FF mode, AOI0 can be J(A) & ~Q(B) | ~K(C) & Q(B), and AOI1 is CLK(D). so, the configuration could be:

AOI0[IN0A_CFG] = 01; AOI0[IN0B_CFG] = 10; AOI0[IN0C_CFG] = 11; AOI0[IN0D_CFG] = 11, that is In0= A & ~B.

AOI0[IN1A_CFG] = 11; AOI0[IN1B_CFG] = 01; AOI0[IN1C_CFG] = 10; AOI0[IN1D_CFG] = 11, that is In1 = ~C & B.

AOI0[IN2A_CFG] = 00; AOI0[IN2B_CFG] = 00; AOI0[IN2C_CFG] = 00; AOI0[IN2D_CFG] = 00, that is In2 = 0.

AOI0[IN3A_CFG] = 00; AOI0[IN3B_CFG] = 00; AOI0[IN3C_CFG] = 00; AOI0[IN3D_CFG] = 00, that is In3= 0.

In this case, B is replaced, therefore we set FB_OVRD as 01.

In this mode, suggest enabling the input sync or filter function to remove the possible glitch.

#### *45.6.2.6 Latch Mode*

When register bits CTRL[MOD] are configured as 0x5, Latch mode will be selected.

In this mode, AOI0 is connected to LATCH D port, AOI1 is connected to LATCH CLK port. Different from D-FF mode, in Latch mode, D port will be passed only when CLK is high, and output will be kept when CLK is low.

LU_OUTB is always the complement of LU_OUTA. In this mode, suggest enabling the input sync or filter function to remove the possible glitch.

### 45.6.3 LU Timing Between Inputs and Outputs

For the modes with input sync and Filter function enabled, the delay between input and output will be:

- Sync Delay: 2
- Flip-Flop Delay: 1
- Filter Delay: (FILT[CNT] + 3) x FILT[PER] + 2
- Whole Delay = ((FILT[CNT] + 3) x FILT[PER] + 2) + 3

All delay numbers here are based on the bus clock.

## 45.7 Register Definition

LU base address : 4007_0000h

**Register Overview - LU**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h+n*10h** | LU_AOI_0n (AOI0 Configuration Register) | 32 | RW | 0000_0000h | page |
| **4h+n*10h** | LU_AOI_1n (AOI1 Configuration Register) | 32 | RW | 0000_0000h | page |
| **8h+n*10h** | LU_CTRLn (LG Configuration Register) | 32 | RW | 0000_0000h | page |
| **Ch+n*10h** | LU_FILTn (Input Filter Register) | 32 | RW | 0000_0000h | page |

### 45.7.1 LU_AOI_0n (AOI0 Configuration Register)

Offset : 0h+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IN0A_CFG | | IN0B_CFG | | IN0C_CFG | | IN0D_CFG | | IN1A_CFG | | IN1B_CFG | | IN1C_CFG | | IN1D_CFG | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IN2A_CFG | | IN2B_CFG | | IN2C_CFG | | IN2D_CFG | | IN3A_CFG | | IN3B_CFG | | IN3C_CFG | | IN3D_CFG | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**LU_AOI_0n field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-30**<br>**IN0A_CFG** | RW | In0A Configuration<br>00b - In0A = 0<br>01b - In0A = LU_INPUTnA<br>10b - In0A = ~LU_INPUTnA<br>11b - In0A = 1 |
| **29-28**<br>**IN0B_CFG** | RW | In0B Configuration<br>00b - In0B = 0<br>01b - In0B = LU_INPUTnB<br>10b - In0B = ~LU_INPUTnB<br>11b - In0B = 1 |

| | | |
|---|---|---|
| **27-26**<br>**IN0C_CFG** | RW | In0C Configuration<br>00b - In0C = 0<br>01b - In0C = LU_INPUTnC<br>10b - In0C = ~LU_INPUTnC<br>11b - In0C = 1 |
| **25-24**<br>**IN0D_CFG** | RW | In0D Configuration<br>00b - In0D = 0<br>01b - In0D = LU_INPUTnD<br>10b - In0D = ~LU_INPUTnD<br>11b - In0D = 1 |
| **23-22**<br>**IN1A_CFG** | RW | In1A Configuration<br>00b - In1A = 0<br>01b - In1A = LU_INPUTnA<br>10b - In1A = ~LU_INPUTnA<br>11b - In1A = 1 |
| **21-20**<br>**IN1B_CFG** | RW | In1B Configuration<br>00b - In1B = 0<br>01b - In1B = LU_INPUTnB<br>10b - In1B = ~LU_INPUTnB<br>11b - In1B = 1 |
| **19-18**<br>**IN1C_CFG** | RW | In1C Configuration<br>00b - In1C = 0<br>01b - In1C = LU_INPUTnC<br>10b - In1C = ~LU_INPUTnC<br>11b - In1C = 1 |
| **17-16**<br>**IN1D_CFG** | RW | In1D Configuration<br>00b - In1D = 0<br>01b - In1D = LU_INPUTnD<br>10b - In1D = ~LU_INPUTnD<br>11b - In1D = 1 |
| **15-14**<br>**IN2A_CFG** | RW | In2A Configuration<br>00b - In2A = 0<br>01b - In2A = LU_INPUTnA<br>10b - In2A = ~LU_INPUTnA<br>11b - In2A = 1 |
| **13-12**<br>**IN2B_CFG** | RW | In2B Configuration<br>00b - In2B = 0<br>01b - In2B = LU_INPUTnB<br>10b - In2B = ~LU_INPUTnB<br>11b - In2B = 1 |

| | | |
|---|---|---|
| **11-10**<br>**IN2C_CFG** | RW | In2C Configuration<br>00b - In2C = 0<br>01b - In2C = LU_INPUTnC<br>10b - In2C = ~LU_INPUTnC<br>11b - In2C = 1 |
| **9-8**<br>**IN2D_CFG** | RW | In2D Configuration<br>00b - In2D = 0<br>01b - In2D = LU_INPUTnD<br>10b - In2D = ~LU_INPUTnD<br>11b - In2D = 1 |
| **7-6**<br>**IN3A_CFG** | RW | In3A Configuration<br>00b - In3A = 0<br>01b - In3A = LU_INPUTnA<br>10b - In3A = ~LU_INPUTnA<br>11b - In3A = 1 |
| **5-4**<br>**IN3B_CFG** | RW | In3B Configuration<br>00b - In3B = 0<br>01b - In3B = LU_INPUTnB<br>10b - In3B = ~LU_INPUTnB<br>11b - In3B = 1 |
| **3-2**<br>**IN3C_CFG** | RW | In3C Configuration<br>00b - In3C = 0<br>01b - In3C = LU_INPUTnC<br>10b - In3C = ~LU_INPUTnC<br>11b - In3C = 1 |
| **1-0**<br>**IN3D_CFG** | RW | In3D Configuration<br>00b - In3D = 0<br>01b - In3D = LU_INPUTnD<br>10b - In3D = ~LU_INPUTnD<br>11b - In3D = 1 |

### 45.7.2 LU_AOI_1n (AOI1 Configuration Register)

Offset : 4h+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IN0A_CFG | | IN0B_CFG | | IN0C_CFG | | IN0D_CFG | | IN1A_CFG | | IN1B_CFG | | IN1C_CFG | | IN1D_CFG | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IN2A_CFG | | IN2B_CFG | | IN2C_CFG | | IN2D_CFG | | IN3A_CFG | | IN3B_CFG | | IN3C_CFG | | IN3D_CFG | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**LU_AOI_1n field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-30**<br>**IN0A_CFG** | RW | In0A Configuration<br>00b - In0A = 0<br>01b - In0A = LU_INPUTnA<br>10b - In0A = ~LU_INPUTnA<br>11b - In0A = 1 |
| **29-28**<br>**IN0B_CFG** | RW | In0B Configuration<br>00b - In0B = 0<br>01b - In0B = LU_INPUTnB<br>10b - In0B = ~LU_INPUTnB<br>11b - In0B = 1 |
| **27-26**<br>**IN0C_CFG** | RW | In0C Configuration<br>00b - In0C = 0<br>01b - In0C = LU_INPUTnC<br>10b - In0C = ~LU_INPUTnC<br>11b - In0C = 1 |
| **25-24**<br>**IN0D_CFG** | RW | In0D Configuration<br>00b - In0D = 0<br>01b - In0D = LU_INPUTnD<br>10b - In0D = ~LU_INPUTnD<br>11b - In0D = 1 |
| **23-22**<br>**IN1A_CFG** | RW | In1A Configuration<br>00b - In1A = 0<br>01b - In1A = LU_INPUTnA<br>10b - In1A = ~LU_INPUTnA<br>11b - In1A = 1 |
| **21-20**<br>**IN1B_CFG** | RW | In1B Configuration<br>00b - In1B = 0<br>01b - In1B = LU_INPUTnB<br>10b - In1B = ~LU_INPUTnB<br>11b - In1B = 1 |

| | | |
|---|---|---|
| **19-18**<br>**IN1C_CFG** | RW | In1C Configuration<br>00b - In1C = 0<br>01b - In1C = LU_INPUTnC<br>10b - In1C = ~LU_INPUTnC<br>11b - In1C = 1 |
| **17-16**<br>**IN1D_CFG** | RW | In1D Configuration<br>00b - In1D = 0<br>01b - In1D = LU_INPUTnD<br>10b - In1D = ~LU_INPUTnD<br>11b - In1D = 1 |
| **15-14**<br>**IN2A_CFG** | RW | In2A Configuration<br>00b - In2A = 0<br>01b - In2A = LU_INPUTnA<br>10b - In2A = ~LU_INPUTnA<br>11b - In2A = 1 |
| **13-12**<br>**IN2B_CFG** | RW | In2B Configuration<br>00b - In2B = 0<br>01b - In2B = LU_INPUTnB<br>10b - In2B = ~LU_INPUTnB<br>11b - In2B = 1 |
| **11-10**<br>**IN2C_CFG** | RW | In2C Configuration<br>00b - In2C = 0<br>01b - In2C = LU_INPUTnC<br>10b - In2C = ~LU_INPUTnC<br>11b - In2C = 1 |
| **9-8**<br>**IN2D_CFG** | RW | In2D Configuration<br>00b - In2D = 0<br>01b - In2D = LU_INPUTnD<br>10b - In2D = ~LU_INPUTnD<br>11b - In2D = 1 |
| **7-6**<br>**IN3A_CFG** | RW | In3A Configuration<br>00b - In3A = 0<br>01b - In3A = LU_INPUTnA<br>10b - In3A = ~LU_INPUTnA<br>11b - In3A = 1 |
| **5-4**<br>**IN3B_CFG** | RW | In3B Configuration<br>00b - In3B = 0<br>01b - In3B = LU_INPUTnB<br>10b - In3B = ~LU_INPUTnB<br>11b - In3B = 1 |

| | | |
|---|---|---|
| **3-2**<br>**IN3C_CFG** | RW | In3C Configuration<br>00b - In3C = 0<br>01b - In3C = LU_INPUTnC<br>10b - In3C = ~LU_INPUTnC<br>11b - In3C = 1 |
| **1-0**<br>**IN3D_CFG** | RW | In3D Configuration<br>00b - In3D = 0<br>01b - In3D = LU_INPUTnD<br>10b - In3D = ~LU_INPUTnD<br>11b - In3D = 1 |

### 45.7.3 LU_CTRLn (LG Configuration Register)

Offset : 8h+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | BYPASS | | SYNC | | | | FB_OVRD | | 0 | MOD | | | INIT_EN | FF_INIT |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**LU_CTRLn field descriptions**

| Field | Type | Description |
|---|---|---|
| **13-12**<br>**BYPASS** | RW | Force Bypass Control<br>BYPASS[0]: control bypass Flip-Flop and route the AOI_0(Filter_0) value directly to LU_OUTA<br>BYPASS[1]: control bypass Flip-Flop and route the AOI_1(Filter_0) value directly to LU_OUTB<br><br>0b - Will not force the bypass<br>1b - Whatever "MOD" is, will force bypass Flip-Flop |

| | | |
|---|---|---|
| **11-8**<br>**SYNC** | RW | LG Input Synchronous Control<br>SYNC[0]: control input "LU_INPUTnA" will be synced by two bus clk cycles or not<br>SYNC[1]: control input "LU_INPUTnB" will be synced by two bus clk cycles or not<br>SYNC[2]: control input "LU_INPUTnC" will be synced by two bus clk cycles or not<br>SYNC[3]: control input "LU_INPUTnD" will be synced by two bus clk cycles or not<br><br>0b - Input will not be synced.<br>1b - Input will be synced by two bus clock cycles |
| **7-6**<br>**FB_OVRD** | RW | LG Output Feedback Override Control<br>When "MOD" is configured as JK-FF mode, it requires the flip-flop output feedback to LG input and replace one of the four inputs.<br>00b - Replace LU_INPUTnA<br>01b - Replace LU_INPUTnB<br>10b - Replace LU_INPUTnC<br>11b - Replace LU_INPUTnD |
| **4-2**<br>**MOD** | RW | Flip-Flop Mode Configuration<br>000b - Bypass mode(Default)<br>001b - RS trigger mode<br>010b - T-FF mode<br>011b - D-FF mode<br>100b - JK-FF mode<br>101b - Latch mode<br>110b - Reserved<br>111b - Reserved |
| **1**<br>**INIT_EN** | RW | Flip-flop Initial Output Enable Control<br>Force the value of FF_INIT to be presented on flip-flop output, this bit is auto clear to 0.<br>0b - Write 0 does not generate enable pulse<br>1b - Write 1 will generate enable pulse<br>This bit should be set after FF_INIT is set. |
| **0**<br>**FF_INIT** | RW | Configure Flip-flop Initial Value<br>FF_INIT does not take effect until "1" is written into INIT_EN<br>0b - Configure the output of flip-flop as "0"<br>1b - Configure the output of flip-flop as "1"<br>This bit should be set before INIT_EN set. |

### 45.7.4 LU_FILTn (Input filter Register)

Offset : Ch+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | CNT0 | | | PRE0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | CNT1 | | | PRE1 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**LU_FILTn field descriptions**

| Field | Type | Description |
|---|---|---|
| **26-24**<br>**CNT0** | RW | Input Filter Sample Count for AOI0<br>These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. A value of 0x0 represents 3 samples. A value of 0x7 represents 10 samples. The value of FILT_CNT affects the input latency. |
| **23-16**<br>**PRE0** | RW | Input Filter Sample Period for AOI0<br>These bits represent the sampling period (in IP bus clock cycles) of the input signals. Each input is sampled multiple times at the rate specified by this field. If FILT_PER is 0x00 (default), then the input filter sample count is bypassed. The value of FILT_PER affects the input latency. When changing values for FILT_PER from one non-zero value to another non-zero value, write a value of zero first to clear the filter. |
| **10-8**<br>**CNT1** | RW | Input Filter Sample Count for AOI1<br>These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. A value of 0x0 represents 3 samples. A value of 0x7 represents 10 samples. The value of FILT_CNT affects the input latency. |
| **7-0**<br>**PRE1** | RW | Input Filter Sample Period for AOI1<br>These bits represent the sampling period (in IP bus clock cycles) of the input signals. Each input is sampled multiple times at the rate specified by this field. If FILT_PER is 0x00 (default), then the input filter sample count is bypassed. The value of FILT_PER affects the input latency. When changing values for FILT_PER from one non-zero value to another non-zero value, write a value of zero first to clear the filter. |

# Chapter 46 Analog-to-Digital Converter (ADC)

## 46.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 46.2 Chip-specific Information

N/A.

## 46.3 Features

The following are the features of the Analog-to-Digital Converter (ADC) module:

- High performance
  - 12-bit, 10-bit, or 8-bit configurable resolution.
  - Programmable sampling time for each channel.
  - Data alignment with built-in data coherency.
  - DMA support.
  - Average out of maximum 32 conversions is supported.
- Low power
  - Can work in MCU WAIT/STOP mode.
  - Wait mode, prevents ADC overrun in applications with low bus clock frequency.
  - Auto disable mode, ADC is automatically disabled except during the active conversion phase. This dramatically reduces the power consumption of the ADC.
- Analog input channels
  - 32 external analog inputs from IO.
  - 6 internal channels (one for temp-sensor).
- Start-of-conversion can be initiated:
  - By software.
  - By hardware triggers with configurable polarity.
- Conversion modes
  - Can convert a single channel or can scan a sequence of channels.
  - Single mode converts selected inputs once per trigger.
  - Continuous mode converts selected inputs continuously.
  - Discontinuous mode.
  - Sequence group mode.
- Interrupt generation
  - At the end of sampling, end of conversion, and end of sequence conversion.
  - In the case of ADC compare or overrun events.
  - When ADC is ready, or FIFO is ready.
- ADC compare function
- Converter uses an internal reference or an external reference
- Single-ended and differential-input configurations available for each sequence step.
- Gain/offset compensation support.

***Note:***

1) Only 32-bit access is supported.
2) Sequence group mode, single-ended/differential-input configurations available for each sequence step and gain/offset compensation features are only supported for FC7240

Flagchip

## 46.4 Overview

This ADC is a successive approximation (SAR) analog-to-digital converter with 12-bit accuracy. It has up to 38 channels allowing it to measure signals from 32 external and 6 internal analog sources (including one for temp-sensor). A/D conversion of the various channels can be performed in a single, continuous, discontinuous, and group mode. The results data are stored in FIFO or results register and the format can be left-aligned or right-aligned.

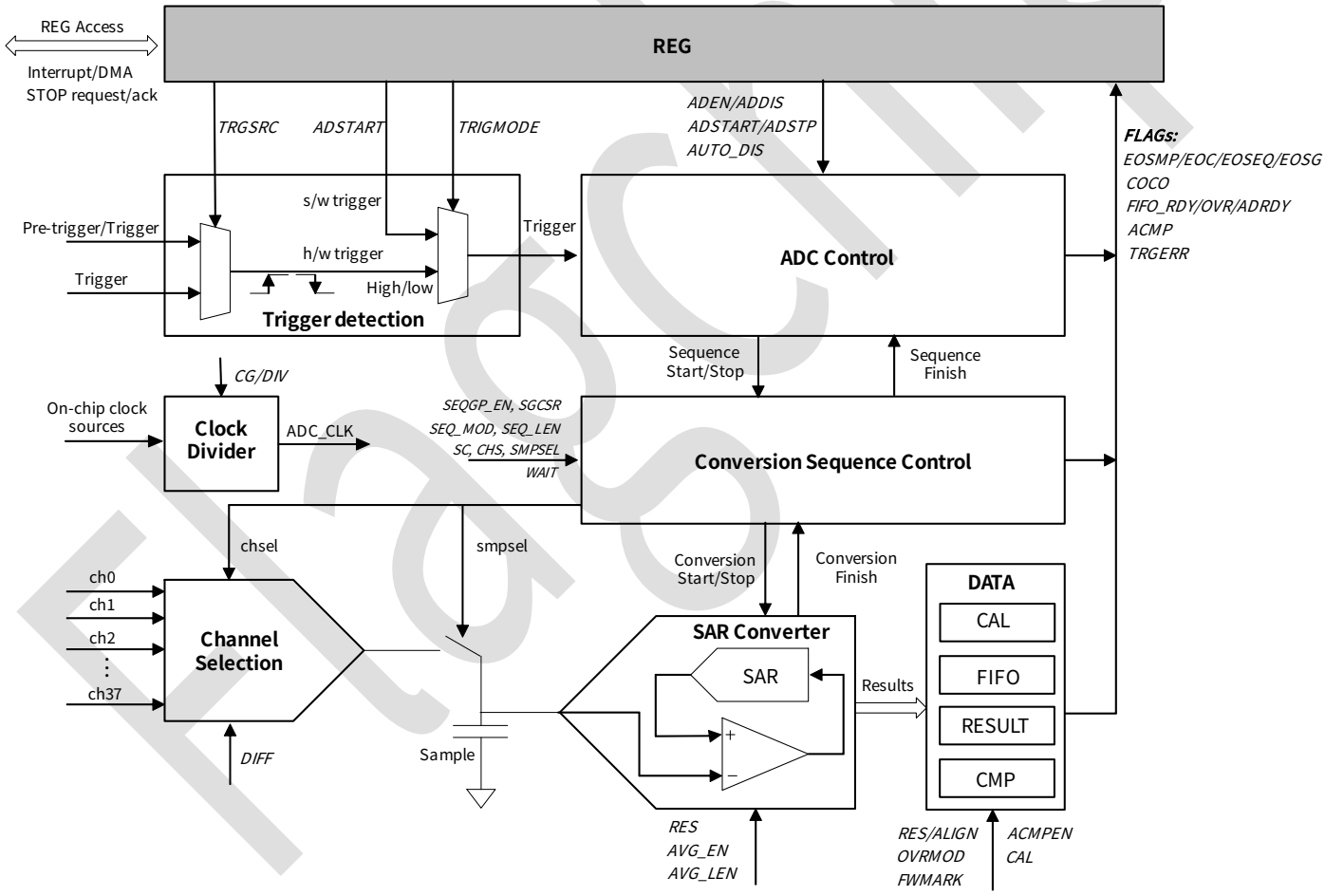
FC7240F2M supports 2 ADCs, and it supports selection of an internal reference or an external reference by ADC_CFG2[REF_EXT]. The references are connected as follows:

- External Reference: VREFH and VREFL
- Internal Reference: VDDA and VSSA

## 46.5 Block Diagram

The following is the block diagram of the ADC module.

**Figure 71. ADC block diagram**




## 46.6 Functional Description

### 46.6.1 ADC Control

#### *46.6.1.1 ADC On-off Control (ADEN, ADDIS)*

At MCU power-up, the ADC is disabled and put in power-down mode (ADEN=0). The ADC needs a stabilization time of Tsu before it starts converting accurately.

Two control bits are used to enable or disable the ADC:

Set ADEN = 1 to enable the ADC. The ADRDY flag is set as soon as the ADC is ready for operation.

- Set ADDIS = 1 to disable the ADC and put the ADC in power down mode. The ADEN and ADDIS bits are then automatically cleared by hardware as soon as the ADC is fully disabled.
- Conversion can then start either by setting ADSTART=1 or when an external trigger event occurs if triggers are enabled.

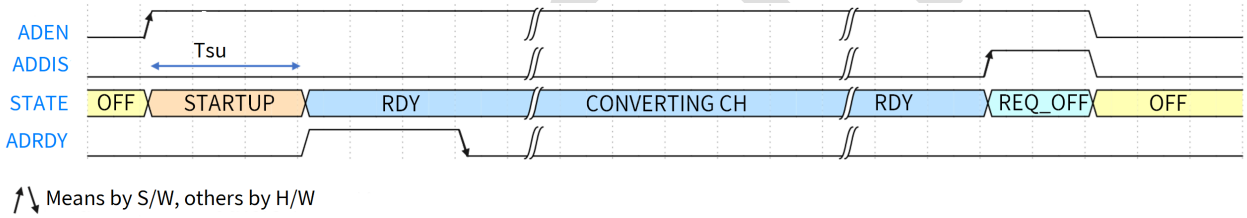
Follow this procedure to enable the ADC

- Set ADEN = 1 in the ADC_CONTROL register.
- Wait until ADRDY = 1 in the ADC_INT_STATUS register (ADRDY is set after the ADC startup time). This can be handled by interrupt if the interrupt is enabled by setting the ADRDYIE bit in the ADC_INT_STATUS register.

Follow this procedure to disable the ADC

- Check that ADSTART = 0 in the ADC_CONTROL register to ensure that no conversion is ongoing. If required, stop any ongoing conversion by writing 1 to the ADSTP bit in the ADC_CONTROL register and waiting until this bit is read at 0.
- Set ADDIS = 1 in the ADC_CONTROL register.
- If required by the application, wait until ADEN=0 in the ADC_CONTROL register, indicating that the ADC is fully disabled (ADDIS is automatically reset once ADEN = 0).

**Figure 72. ADC ON/OFF control**




### *46.6.1.2 Configuring the ADC*

Software must write to the ADEN bit in the ADC_CONTROL register if the ADC is disabled (ADEN must be 0).

Software must only write to the ADSTART and ADDIS bits in the ADC_CONTROL register only if the ADC is enabled in the ready state (RDY) and there is no pending request to disable the ADC (ADEN = 1 and ADDIS = 0).

For all the other control bits in the ADC_CFG1, ADC_CFG2, ADC_SMPR, ADC_CMP_CTRL, ADC_CMP_TR, ADC_SGCSR_n and ADC_SC_n registers, software must only write to the configuration control bits if there is no conversion ongoing (ADSTART = 0). This means at the time after the ADRDY flag is got.

### *46.6.1.3 Channel Selection (ADC_SC_n[CHS])*

There are up to 38 multiplexed channels:

- 32 analog inputs from IO
- 6 internal analog inputs (including one for temp-sensor).

It is possible to convert a single channel or to automatically scan a sequence of channels.

In single conversion mode, continuous conversion mode, discontinuous mode 0, the sequence of the channels to be converted is organized as ADC_SC_0[CHS], then ADC_SC_1[CHS], ..., then ADC_SC_(SEQ_LEN)[CHS]. The sequence length is programmed in SEQ_LEN of ADC_CFG1.

In discontinuous mode 1, the sequence of the channels to be converted is defined by pre-triggers. One pre-trigger(n) event will select ADC_SC_n[CHS] as the input channel, and one or more pre-triggers specify a conversion sequence on selected channels.

In sequence group mode, the sequence of the channels to be converted in the group is organized as ADC_SC_(SG_START)[CHS], then ADC_SC_(SG_START+1) [CHS], ..., then ADC_SC_(SG_END) [CHS]. The sequence length is decided by SG_START and SG_END.

ADC_SC_n[DIFF] can be used to select differential-input or single-ended configurations for the channel specified in sequence step n. If the channel does not support differential-input, a single-ended configuration will be selected even if ADC_SC_n[DIFF] is set to 1.

**Table 66. Channel decode**

| ADC_SC_n[CHS] | Channel Select (single-ended) | Channel Select (differential) |
|---|---|---|
| 6'b000000 | Ext channel 0 | Ext channel 0, Ext channel 4 |
| 6'b000001 | Ext channel 1 | Ext channel 1, Ext channel 5 |
| 6'b000010 | Ext channel 2 | Ext channel 2, Ext channel 6 |
| 6'b000011 | Ext channel 3 | Ext channel 3, Ext channel 7 |
| 6'b000100 | Ext channel 4 | Reserved |
| 6'b000101 | Ext channel 5 | Reserved |
| 6'b000110 | Ext channel 6 | Reserved |
| 6'b000111 | Ext channel 7 | Reserved |
| 6'b001000 | Ext channel 8 | Reserved |
| 6'b001001 | Ext channel 9 | Reserved |
| 6'b001010 | Ext channel 10 | Reserved |
| 6'b001011 | Ext channel 11 | Reserved |
| 6'b001100 | Ext channel 12 | Reserved |
| 6'b001101 | Ext channel 13 | Reserved |
| 6'b001110 | Ext channel 14 | Reserved |
| 6'b001111 | Ext channel 15 | Reserved |
| 6'b010000 | Ext channel 16 | Reserved |
| 6'b010001 | Ext channel 17 | Reserved |
| 6'b010010 | Ext channel 18 | Reserved |
| 6'b010011 | Ext channel 19 | Reserved |
| 6'b010100 | Ext channel 20 | Reserved |
| 6'b010101 | Ext channel 21 | Reserved |
| 6'b010110 | Ext channel 22 | Reserved |
| 6'b010111 | Ext channel 23 | Reserved |
| 6'b011000 | Ext channel 24 | Reserved |
| 6'b011001 | Ext channel 25 | Reserved |
| 6'b011010 | Ext channel 26 | Reserved |
| 6'b011011 | Ext channel 27 | Reserved |
| 6'b011100 | Ext channel 28 | Reserved |
| 6'b011101 | Ext channel 29 | Reserved |
| 6'b011110 | Ext channel 30 | Reserved |
| 6'b011111 | Ext channel 31 | Reserved |
| 6'b100000 | Int channel 0 | Reserved |

| ADC_SC_n[CHS] | Channel Select (single-ended) | Channel Select (differential) |
|---|---|---|
| 6'b100001 | Int channel 1 | Reserved |
| 6'b100010 | Int channel 2 | Reserved |
| 6'b100011 | Int channel 3 | Reserved |
| 6'b100100 | Int channel 4 | Reserved |
| 6'b100101 | Reserved | Int channel 5 (temp-sensor) |

#### *46.6.1.4 Sampling Time Selection (ADC_SC_n[SMPSEL])*

Before starting a conversion, the ADC needs to establish a direct connection between the voltage source to be measured and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the sample and hold the capacitor to the input voltage level.

Having a programmable sampling time allows trimming the conversion speed according to the input resistance of the input voltage source. The ADC channel ADC_SC_n[CHS] samples the input voltage for several ADC clock cycles that can be selected using the ADC_SC_n[SMPSEL] bits and ADC_SMPR register.

The ADC indicates the end of the sampling phase by setting the EOSMP flag.

#### *46.6.1.5 Starting Conversions (ADSTART)*

Software starts ADC conversions by setting ADSTART = 1.

When ADSTART is set, the conversion:

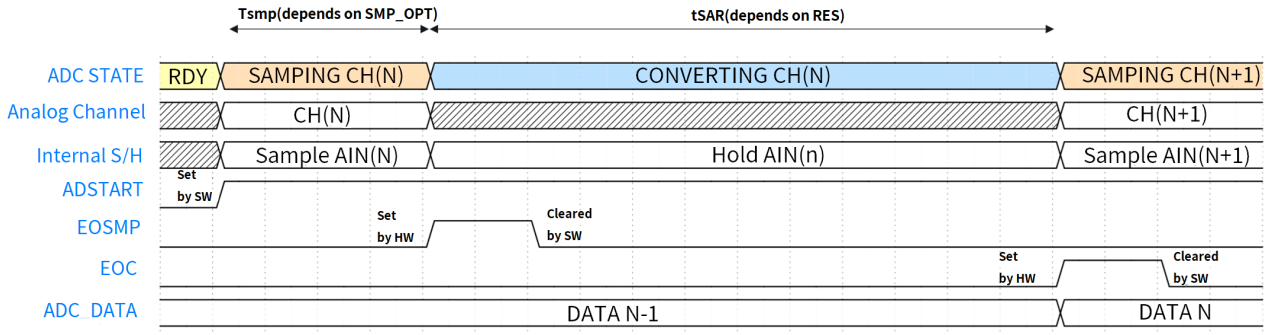
- Starts immediately if TRIGMODE = 0x0 (software trigger)
- At the next active edge of the selected hardware trigger if TRIGMODE ≠ 0x0

The ADSTART bit is cleared by hardware:

- In single mode (SEQ_MOD = 0) with software trigger:
  - At any end of the conversion sequence (EOSEQ = 1).
- In discontinuous mode 0 (SEQ_MOD = 2) with software trigger:
  - At any end of conversion.
- In all cases
  - After execution of the ADSTP procedure invoked by software.

Any time when reasserting ADSTART, make sure the ADRDY flag is got.

**Figure 73. ADC start sequence**



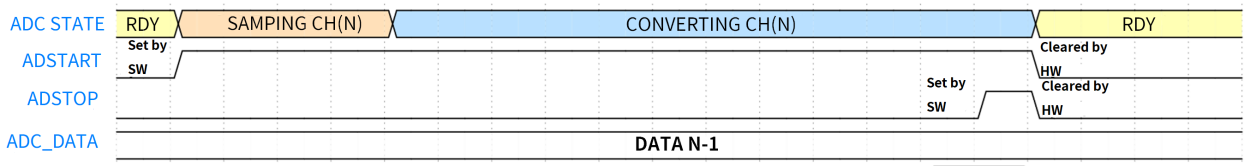

#### *46.6.1.6 Stopping an Ongoing Conversion (ADSTP)*

The software can decide to stop any ongoing conversions by setting ADSTP=1 in the ADC_CONTROL register. This will reset the ADC operation and the ADC will be idle, ready for a new operation.

When the ADSTP bit is set by software, any ongoing conversion is aborted. The scan sequence is also aborted and reset (meaning that restarting the ADC would starter-start a new sequence).

Once this procedure is complete, the ADC will back to the ready state and the ADSTP and ADSTART bits are both cleared by hardware, the software must wait until ADSTP = 0 before starting new conversions.

**Figure 74. ADC stop sequence**




### 46.6.2 Trigger Detection

#### *46.6.2.1 Conversion on External Trigger and Trigger Polarity (TRIGMODE, TRIGSRC)*

A conversion or a sequence of conversions can be triggered either by software or by an external event. If the TRIGMODE control bits are not equal to "0," then external events can trigger a conversion with the selected polarity. The trigger selection is effective once the software has set bit ADSTART = 1.

Any hardware triggers that occur while a conversion is ongoing are ignored.

If bit ADSTART=0, any hardware triggers that occur are ignored.

The below table provides the correspondence between the TRIGMODE values and the trigger polarity.

**Table 67. Configuring the trigger polarity**

| TRIGMODE[2:0] | Source |
|---|---|
| 3'b000 | Trigger detection disabled, software trigger |
| 3'b001 | Detection on the rising edge |
| 3'b010 | Detection on falling edge |
| 3'b011 | Detection on both rising and falling edges |
| 3'b100 | Detection on high-level voltage |
| 3'b101 | Detection on low-level voltage |
| 3'b110 | Reserved |
| 3'b111 | Reserved |

TRIGSRC selects a trigger source. The TRIGSRC MUX is before TRIGMODE MUX. These bits are used to select which of 2 possible events can trigger conversions.

**Table 68. Configuring the trigger source**

| TRIGSRC[2:0] | Source |
|---|---|
| 3'b000 | Trigger source from PTIMER/TRGSEL with SCM (ONLY USED for SEQ_MOD=3) |
| 3'b001 | Reserved |
| 3'b010 | Trigger source from TRGSEL (ONLY USED for SEQ_MOD=0-2) |
| 3'b011 | Trigger source from TRIG_LATCH_UNIT (ONLY USED for SEQGP_EN=1) |
| 3'b100 | Reserved |
| 3'b101 | Reserved |
| 3'b110 | Reserved |

| | |
|---|---|
| 3'b111 | Reserved |

### *46.6.2.2 Trigger Sources Selection and Trigger Latch Unit*

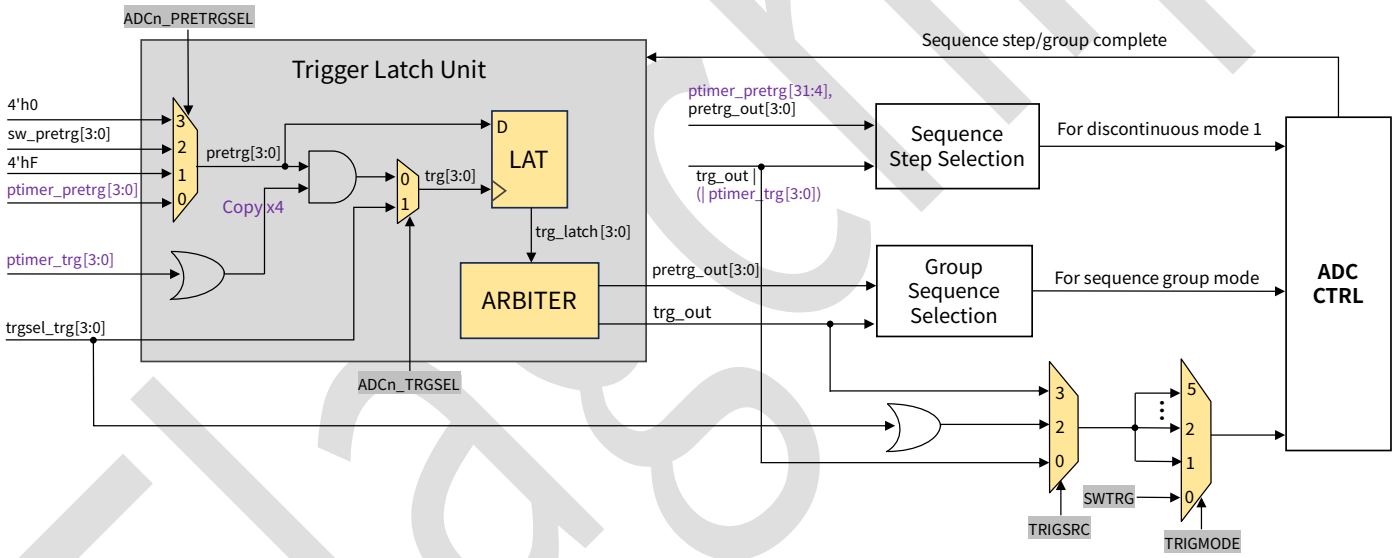
ADC pre-triggers and triggers come from PTIMER/TRGSEL with configuration by the System Control Module (SCM).

In discontinuous mode 1 (SEQ_MOD=3, SEQGP_EN=0), the TRIGSRC register value is omitted and the pre-triggers/ trigger source from PTIMER (ptimer_pretrg[31:0], ptimer_trg[3:0]), TRGSEL with SCM (sw_pretrg[3:0] or 4'hF, trgsel_trg[3:0]) is selected (just like TRIGSRC=0). The TRIGMODE register value is omitted and detection on the rising edge (just like TRIGMODE=1) is selected.

In sequence group mode (SEQGP_EN=1), the TRIGSRC register value is omitted and the pre-triggers/ trigger source from Trigger Latch Unit is used (just like TRIGSRC=3). In this case, PTIMER (ptimer_pretrg[3:0], ptimer_trg[3:0]), TRGSEL with SCM (sw_pretrg[3:0] or 4'hF, trgsel_trg[3:0]) can be selected in Trigger Latch Unit with latch and arbiter function. TRIGMODE register value is omitted and detection on the rising edge (just like TRIGMODE=1) is selected.

In other conversion modes (SEQ_MOD=0/1/2, SEQGP_EN=0), the TRIGSRC register must be configured as 2 to select triggers from TRGSEL (trgsel_trg[3:0]), no pre-trigger is needed.

**Figure 75. ADC pre-trigger and trigger connection**




Trigger requests which are handled by Trigger Latch Unit can be latched and then serviced. These trigger requests can be one-hot or simultaneous, but they will be processed on a round-robin or fix-priority basis which is controlled by TRG_PRI. The latched triggers are processed by ADC one at a time, the next one can only be processed when the current sequence step or group is completed. When using the trigger latch feature, the bus clock frequency must be greater than 1/3 the frequency of the ADC function clock (ADC_CLK).

Trigger Latch Unit accept triggers with pre-triggers from PTIMER (ptimer_pretrg[3:0], ptimer_trg[3:0]) when ADCn_PRETRGSEL=0 and ADCn_TRGSEL=0. ptimer_pretrg[3:0] is one-hot and when any bit of ptimer_pretrg[3:0] is set , ptimer_trg[0] will be set too, the pretrg[3:0] will be latched by trg[3:0] and then serviced.

Trigger Latch Unit accepts triggers with pre-triggers from TRGSEL with SCM (4'hF is used as pre-triggers, trgsel_trg[3:0] is used as triggers) when ADCn_PRETRGSEL =1 and ADCn_TRGSEL =1. trgsel_trg[3:0] can be one-hot or simultaneous and when one or more bits of trgsel_trg[3:0] are set, trg[3:0] will be latched (use trg[3:0] to latch 4'hF, so only the bits in trg[3:0] which are active will be latched) and then serviced.

Trigger Latch Unit accepts triggers with pre-triggers from TRGSEL with SCM (sw_pretrg[3:0] is used as pre-triggers, trgsel_trg[3:0] is used as triggers) when ADCn_PRETRGSEL =2 and ADCn_TRGSEL =1. sw_pretrg[3:0] is one-hot and so must

be used with related trgsel_trg[3:0] bit, such as when using sw_pretrg[1] as a pre-trigger, only trgsel_trg[1] can be used as a trigger, the pretrg[3:0] will be latched by trg[3:0] and then serviced.

Trigger error bits TRGERR in ADC_INT_STATUS register will be set if a new trigger request appears again on the same Trigger Latch Unit input channel while the last one is being latched or processed, an interrupt request can be generated when the related interrupt enable TRG_PRI is set. The user can check bits TRG_STATUS in ADC_INT_STATUS register to see which triggers are latched and can clear these latched triggers by setting TRG_CLR bit in ADC_CFG2 register.

The pre-triggers and triggers from PTIMER or from Trigger Latch Unit after latch and arbitration can be used to select sequence steps (SEQ_MOD=3, SEQGP_EN=0) or group sequences (SEQGP_EN=1) and trigger ADC conversion start.

### 46.6.3 Conversion Sequence Mode

#### *46.6.3.1 Single Conversion Mode*

In single conversion mode, the ADC converts the sequence once, the sequence length is defined by SEQ_LEN and each sequence step is defined in SC_n (n is from 0 to SEQ_LEN) register. Conversion is started by either:

- Setting the ADSTART bit in the ADC_CONTROL register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the FIFO
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOSEQ (end of sequence) flag is set
- An interrupt is generated if the EOSEQIE bit is set

Then the ADC stops until a new external trigger event occurs or the ADSTART bit is set again.

**Figure 76. Single conversion mode by software trigger**

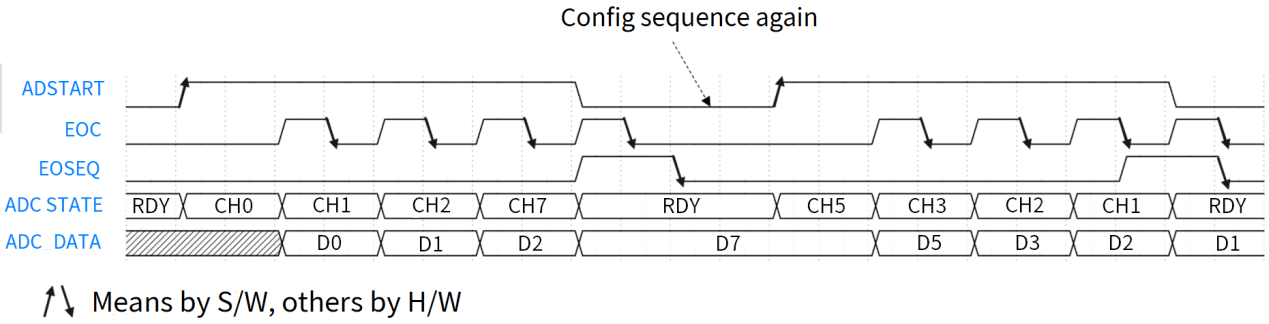
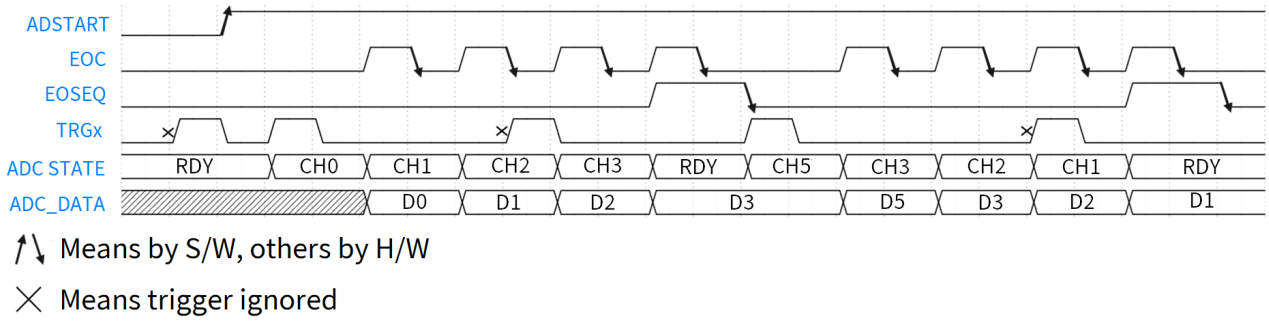



**Figure 77. Single conversion mode by hardware trigger**



#### 46.6.3.2 Continuous Conversion Mode

In continuous conversion mode, when a software or hardware trigger event occurs, the ADC performs a sequence of conversions, converting the sequence once and then automatically restarts and continuously performs the same sequence of conversions. The sequence length is defined by SEQ_LEN and each sequence step is defined in SC_n (n is from 0 to SEQ_LEN) register. Conversion is started by either:

- Setting the ADSTART bit in the ADC_CONTROL register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the FIFO
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOSEQ (end of sequence) flag is set
- An interrupt is generated if the EOSEQIE bit is set

Then, a new sequence restarts immediately, and the ADC continuously repeats the conversion sequence.

**Figure 78. Continuous conversion mode by software trigger**

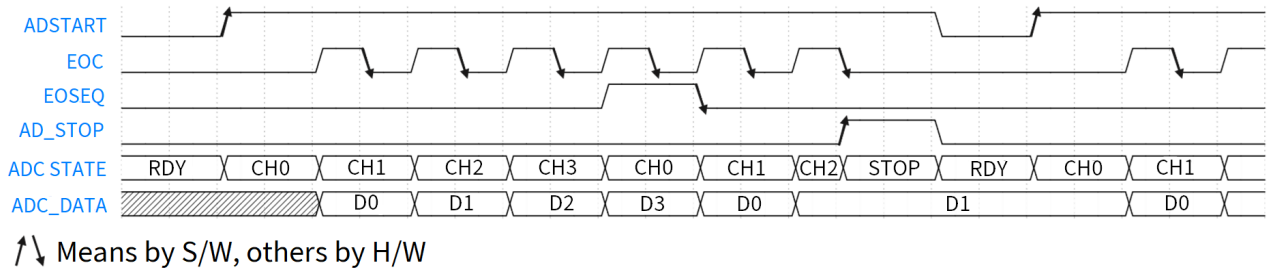
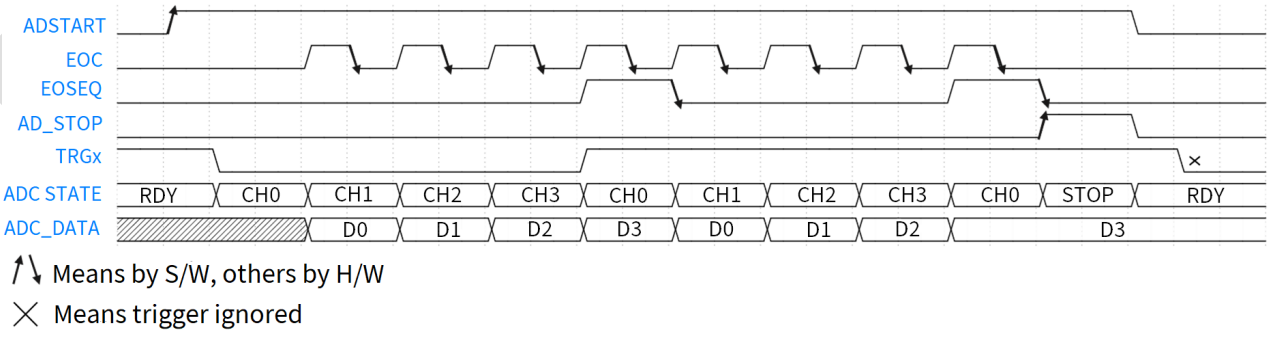



**Figure 79. Continuous conversion mode by hardware trigger**




#### 46.6.3.3 Discontinuous Mode 0

In this mode, each sequence step needs a hardware or software trigger event, the sequence length is defined by SEQ_LEN and each sequence step is defined in SC_n (n is from 0 to SEQ_LEN) register. The sequence will repeat if there is a new trigger event coming and can only be stopped when ADSTP is set.

Example:

Channels to be converted = 0, 3, 7, 10 (SEQ_LEN=3)

- 1st trigger: channel 0 is converted and an EOC event is generated

- 2nd trigger: channel 3 is converted and an EOC event is generated
- 3rd trigger: channel 7 is converted and an EOC event is generated
- 4th trigger: channel 10 is converted and both EOC and EOSEQ events are generated.
- 5th trigger: channel 0 is converted an EOC event is generated
- 6th trigger: channel 3 is converted and an EOC event is generated

#### *46.6.3.4 Discontinuous Mode 1*

In this mode, hardware trigger events with pre-triggers are required to start each conversion step defined in the sequence. And conversion will stop when ADSTP is set.

The ADC can work with the PTIMER pre-trigger back-to-back (BTB) operation enabled. With the PTIMER BTB operation, the ADC can sample the specified channel with a configurable delay time. For example, the ADC will select the channel specified in ADC_SC_n[CHS], where n is determined by PTIMER pre-trigger. SEQ_LEN is invalid for this mode.

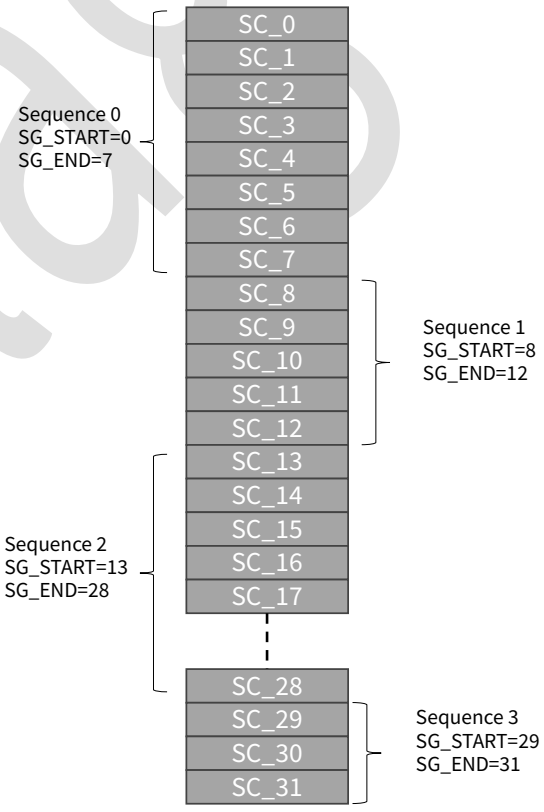
In this mode, the ADC conversion results are stored in the ADC_RESULT_n register instead of the FIFO.

#### *46.6.3.5 Sequence Group Mode*

In single conversion mode/ continuous conversion mode/ discontinuous mode 0/ discontinuous mode 1, only one kind of conversion sequence is supported. While in sequence group mode, a group of (max 4) conversion sequences are supported. Sequence group mode is selected when SEQGP_EN=1.

In sequence group mode, one sequence in the group is defined by sequence start point (SGCSRn.SG_START) and sequence end point (SGCSRn.SG_END). Each sequence step is defined in SC_m (m is from SGCSRn.SG_START to SGCSRn.SG_END) register. Up to 4 kinds of pre-defined (by SGCSRn) sequences are supported in the group mode, and the sequences in the group are selected by pre-triggers(pretrg_out[3:0]) and trigger (trg_out) from Trigger Latch Unit.

**Figure 80. Sequence defined from SC_n register**

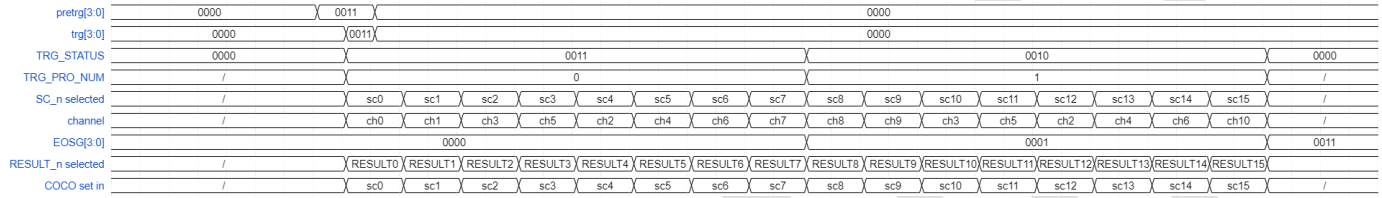



Sequence n (defined by SGCSRn) is selected when pretrg_out[n] and trg_out is active. ADC will start to convert the sequence steps defined in SC_m (m is from SGCSRn.SG_START to SGCSRn.SG_END), once all the sequence steps are finished, an end of

sequence group flag SGCSRn.EOSG will be set, an interrupt can be generated if the related interrupt enable SGCSRn.EOSGIE is set. The end of sequence group will also be sent to Trigger Latch Unit for the next trigger request processing. During the sequence, the step m conversion result is saved to the related RESULT_m register and the related conversion complete flag is set in SC_m.COCO.

Here is a waveform that shows the conversion sequences in the group when sequence group mode is selected. In this case, there are two pre-trigger and trigger requests that are latched and served, pretrg[0]/ trg[0] are served first and then pretrg[1]/ trg[1]. pretrg[0]/ trg[0] selects pre-defined sequence 0 and pretrg[1]/ trg[1] selects pre-defined sequence 1. Pre-defined sequence 0 is defined by SGCSR0.SG_START(=0) to SGCSR0.SG_END(=7) with conversion step configurations in SC_0, SC_1…SC_7. Pre-defined sequence 1 is defined by SGCSR1.SG_START(=8) to SGCSR1.SG_END(=15) with conversion step configurations in SC_8, SC_9…SC_15.

**Figure 81. Sequence group mode**




Set ADSTP during the sequence will abort the sequence and clear all the latched trigger requests in Trigger Latch Unit. Wait TRG_STATUS until it becomes to 0 which means there is no latched trigger any more.

### 46.6.4 Data Management

#### *46.6.4.1 Programmable Resolution (RES)*

It is possible to obtain faster conversion times (tSAR) by reducing the ADC resolution. The resolution can be configured to be either 12, 10, or 8 bits by programming the RES[1:0] bits in the ADC_CFG1 register. Lower resolution allows faster conversion times for applications where high data precision is not required.

The result of the conversion is always 12 bits wide and any unused LSB bits are read as zeroes.

Lower resolution reduces the conversion time needed for the successive approximation steps.

#### *46.6.4.2 ADC Average Function (AVG)*

A hardware average function can be enabled by setting AVG_EN=1 in the DC_CFG2 register. It supports 4-32 times average by configuring AVG_LEN bits in the ADC_CFG2 register. If AVG_EN=1, the average result will be saved in FIFO or ADC_RESULT registers, the original results are not saved and EOSMP/EOC/EOSEQ flags are only asserted at the last average conversion step. The ADC compare function is based on the averaged result.

#### *46.6.4.3 Data Alignment*

At the end of each conversion (when an EOC event occurs), the result of the converted data is stored in the ADC_FIFO or ADC_RESULT_n registers.

The format of the readout data depends on the configured data alignment and resolution.

The ALIGN bit in the ADC_CFG1 register selects the alignment of the data stored after conversion. Data can be right-aligned (ALIGN=0) or left-aligned (ALIGN=1)

| ALIGN | RES[1:0] | 31 | … | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0x0 | | | | | | | data[11:0] | | | | | | | | | | | |
| | 0x1 | | | | | | | | | data[9:0] | | | | | | | | | |

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0x2 | | | | | | | | | | | data[7:0] | | | | | | | |
| 1 | 0x0 | | | data[11:0] | | | | | | | | | | | | | | | |
| | 0x1 | | | data[9:0] | | | | | | | | | | | | | | | |
| | 0x2 | | | | | | | | | | | data[7:0] | | | | | | | |

For single-ended input configuration, converted data is un-signed data from 0 to the maximum (decided by resolution such as 2^12-1 for 12-bit resolution) with input from the minimum (VREFL/VSSA) to the maximum (VREFH/VDDA).

While for differential-input configuration, converted data is also un-signed data from 0 to the maximum (decided by resolution such as 2^12-1 for 12-bit resolution) with input from the minimum (VREFL-VREFH/VSSA-VDDA) to the maximum (VREFH-VREFL/VDDA-VSSA). When the input is VREFL-VREFH, a data 0 is got from ADC; when the input is VREFH-VREFL, a data 4095 (12-bit resolution) is got from ADC; and when the input is 0v, a data 2048 (12-bit resolution) is got from ADC.

#### *46.6.4.4 ADC Overrun (OVR, OVRMOD)*

The overrun flag (OVR) indicates a data overrun event, when the converted data was not read in time by the CPU or the DMA, before the FIFO is full.

The OVR flag is set in the ADC_INT_STATUS register if the FULL flag is still at '1' at the time when a new conversion completes. An interrupt can be generated if the OVRIE bit is set in the ADC_INT_STATUS register.

When an overrun condition occurs, the ADC keeps operating and can continue to convert unless the software decides to stop and reset the sequence by setting the ADSTP bit in the ADC_CONTROL register.

The OVR flag is cleared by software by writing 1 to it.

It is possible to configure if the data is preserved or overwritten when an overrun event occurs by programming the OVRMOD bit in the ADC_CFG1 register:

- OVRMOD=0
  An overrun event preserves the data register from being overwritten: the old data is maintained, and the new conversion is discarded. If OVR remains at 1, further conversions can be performed but the resulting data is discarded.
- OVRMOD=1
  The data register is overwritten with the last conversion result. If OVR remains at 1, further conversions can be performed, and the FIFO always contains the data from the latest conversion.

#### *46.6.4.5 DMA*

When SEQ_MOD=0-2, once the number of data entries in the FIFO is more than the FWMARK setting and bit DMAEN is set, the ADC will send a request to the DMA. This allows the transfer of the converted data from the FIFO to the destination location selected by software.

When SEQ_MOD=3 or SEQGP_EN=1 and DMAEN is set, once the ADC finishes one conversion and the ADC_SC_n[COCO] is set, the DMA request is set and allows the converted data from ADC_RESULT_n transferred to the destination selected by software.

#### *46.6.4.6 Gain and Offset Calibration*

A calibration register CAL is used for gain and offset calibration, after reset OFFSET and GAIN in CAL register will be reset to factor test value. The OFFSET bits defined an offset which will be removed from the conversion results if CAL_EN=1, it is signed data where OFFSET[11] is the sign. The GAIN bits defined a gain factor (GAIN/2^12) which is multiplied with the

conversion results if CAL_EN=1. If CAL_EN=0, both offset and gain calibration are bypassed. An application can do the calibration test when needed and write back the offset and gain value back to OFFSET and GAIN bits.

### 46.6.5 ADC Interrupts

An interrupt can be generated by any of the following events:

- When the ADC is ready (ADRDY flag)
- When the end of the sampling phase occurs (EOSMP flag)
- End of any conversion (EOC/COCO flag)
- End of a sequence of conversions (EOSEQ/EOSG flag)
- When a data overrun occurs (OVR flag)
- When sampled voltage is out of range (ACMP flag)
- When the FIFO data count is greater than the FIFO watermark (FIFO_RDY flag)
- When Trigger Latch Unit new trigger request has been missed (TRGERR)

### 46.6.6 Low Power Features

#### *46.6.6.1 MCU WAIT/STOP Mode*

The ADC can work in MCU WAIT/STOP mode if the stop mode control bit (CSC0/CSC1/CSC2 STOPMODER0[ADCn]) is 0, a stop ack will be given immediately when a stop request comes. All interrupts can generate wakeup events if related interrupt enables are set (except ADRDY/ EOSG/ TRGERR interrupts), sequence group mode and Trigger Latch Unit can’ t be used in STOP mode because it needs bus clock for normal function.

If the stop mode control bit is 1, the ADC will wait for the current conversion sequence done (for discontinuous mode, wait for the current conversion done) and then be disabled (in sequence group mode the latched trigger request will be cleared) by hardware when a stop request comes. During MCU wakeup, the ADC will be enabled again by hardware to ready state.

#### *46.6.6.2 ADC Wait mode*

ADC wait mode conversion can be used to simplify the software as well as optimize the performance of applications clocked at low frequency where there might be a risk of ADC overrun occurring. When the WAIT bit is set to 1 in the ADC_CFG1 register, a new conversion can start only if the FIFO is not full (Note: This mode is only valid for SEQ_MOD = 0/1/2). This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data. Any hardware triggers that occur while a conversion is ongoing or during the wait time preceding the read access are ignored.

#### *46.6.6.3 Auto Disable Mode*

The ADC has an automatic power management feature which is called auto disable mode and is enabled by setting AUTO_DIS=1 in the ADC_CFG1 register.

When AUTO_DIS = 1, the ADC is always powered off when not converting and automatically re-enabled when a conversion is started (by software or hardware trigger). A startup-time is automatically inserted between the trigger event which starts the conversion and the sampling time of the ADC. The ADC is then automatically disabled once the sequence of conversions is complete. Auto disable mode can cause a dramatic reduction in the power consumption of applications that need relatively few conversions or when conversion requests are timed far enough apart (for example with a low-frequency hardware trigger) to justify the extra power and extra time used for switching the ADC on and off.

AUTO_DIS can be configured when the ADC is in OFF state or in RDY state. When configuring it in OFF state, the ADC enable, start, stop, and disable sequences are the same as described in the previous sessions. When configuring it in RDY state, the ADRDY flag will be set again when the transition finishes.

### 46.6.7 ADC Compare

The ADC compare feature is enabled by setting the ACMPEN bit in the ADC_CMP_CTRL register. It is used to monitor that either one selected channel or all enabled channels remain within a configured voltage range (window).

The ADC compare status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold. These thresholds are programmed in the ADC_CMP_TR register. An interrupt can be enabled by setting the ACMPIE bit in the ADC_INT_ENABLE register. The ACMP flag is cleared by software by writing 1 to it.

### 46.6.8 ADC Clock

The ADC module has a dual clock domain architecture, one is the ADC function clock (ADC_CLK) domain, and another is the bus clock domain. All register accesses are in the bus clock domain and all ADC control logics are in the function clock domain. There is an internal clock divider and a clock gating before the divider in ADC, user can configure DIV bits in the ADC_CFG2 register to get different frequency options of ADC_CLK. Before changing DIV, CG must be set to 1 and wait for CG_ACK = 1, after changing DIV bits, CG must be set to 0 and wait for CG_ACK = 0.

## 46.7 Register Definition

ADC0 base address : 4003_B000h

ADC1 base address : 4003_C000h

**Register Overview - ADC**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | ADC_INT_STATUS (Interrupt Status Register) | 32 | RW | 0000_0040h | page |
| **4h** | ADC_INT_ENABLE (Interrupt Enable Register) | 32 | RW | 0000_0000h | page |
| **8h** | ADC_CONTROL (Control Register) | 32 | RW | 0000_0000h | page |
| **Ch** | ADC_CFG1 (Configuration1 Register) | 32 | RW | 0000_0000h | page |
| **10h** | ADC_CFG2 (Configuration2 Register) | 32 | RW | 1000_00A0h | page |
| **14h** | ADC_SMPR (Sampling Rate Register) | 32 | RW | 8020_0802h | page |
| **18h** | ADC_CMP_CTRL (Compare Control Register) | 32 | RW | 0000_0000h | page |
| **1Ch** | ADC_CMP_TR (Compare Threshold Register) | 32 | RW | 0000_0200h | page |
| **28h+n*4h** | ADC_SGCSRn (Sequence Group Control and Status Register) | 32 | RW | 0000_0000h | page |
| **44h** | ADC_CAL (Calibration Register) | 32 | RW | 0000_1000h | page |
| **4Ch** | ADC_FIFO_DATA (FIFO Data Register) | 32 | RW | 0000_0000h | page |
| **50h+n*4h** | ADC_SCn (Sequence Configuration Register) | 32 | RW | 0000_003Fh | page |
| **D0h+n*4h** | ADC_RESULTn (Result Register) | 32 | RW | 0000_0000h | page |

### 46.7.1 ADC_INT_STATUS (Interrupt Status Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | TRGERR | | | | 0 | | | | TRG_STATUS | | | |
| W | | | | | w1c | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | TRG_PRO_NUM | | 0 | | | | FIFO_RDY | ACMP | EMPTY | FULL | OVR | EOSEQ | EOC | EOSMP | ADRDY |
| W | | | | | | | | | w1c | | | w1c | w1c | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADC_INT_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **27-24**<br>**TRGERR** | W1C | Trigger Request Error<br>Each of these bits indicates that Trigger Latch Unit new trigger request has been missed, because the old request has already been latched or is being serviced.<br>Bit 27: Trigger request 3<br>Bit 26: Trigger request 2<br>Bit 25: Trigger request 1<br>Bit 24: Trigger request 0<br>0b - No error occurred<br>1b - Error occurred |
| **19-16**<br>**TRG_STATUS** | RO | Trigger Status<br>Each of these bits indicates that a Trigger Latch Unit trigger request has been latched.<br>Bit 19: Trigger request 3<br>Bit 18: Trigger request 2<br>Bit 17: Trigger request 1<br>Bit 16: Trigger request 0<br>0b - Trigger not latched<br>1b - Trigger latched |
| **14-13**<br>**TRG_PRO_NUM** | RO | Trigger Process Number<br>Indicates the trigger number in Trigger Latch Unit that is being serviced.<br>TRGPRNUM=11 is valid only if TRG_STATUS[19]=1<br>TRGPRNUM=10 is valid only if TRG_STATUS[18]=1<br>TRGPRNUM=01 is valid only if TRG_STATUS[17]=1<br>TRGPRNUM=00 is valid only if TRG_STATUS[16]=1 |

| | | |
|---|---|---|
| **8**<br>**FIFO_RDY** | RO | FIFO Ready<br>Indicates that the current data number in FIFO is more than the FIFO watermark.<br>0b - FIFO not ready<br>1b - FIFO ready |
| **7**<br>**ACMP** | W1C | ADC Compare Flag<br>This bit is set by hardware when the converted voltage exceeds the values programmed in the ADC_TR register. It is cleared by software writing 1 to it.<br>0b - No ADC compare event occurred (or the flag event was already acknowledged and cleared by software)<br>1b - ADC compare event occurred |
| **6**<br>**EMPTY** | RO | FIFO Empty Status<br>This bit is set by hardware when the FIFO is empty. It is cleared by hardware when FIFO is not empty.<br>0b - FIFO is not empty<br>1b - FIFO is empty |
| **5**<br>**FULL** | RO | FIFO Full Status<br>This bit is set by hardware when the FIFO is full. It is cleared by hardware when FIFO is not full.<br>0b - FIFO is not full<br>1b - FIFO is full |
| **4**<br>**OVR** | W1C | ADC Overrun<br>This bit is set by hardware when an overrun occurs, meaning that a new conversion has been completed while the FULL flag was already set. It is cleared by software writing 1 to it.<br>0b - No overrun occurred (or the flag event was already acknowledged and cleared by software)<br>1b - Overrun has occurred |
| **3**<br>**EOSEQ** | W1C | End of Sequence Flag<br>This bit is set by hardware at the end of the conversion of a sequence. It is cleared by software writing 1 to it.<br>ONLY VALID for SEQ_MOD=0-2.<br>0b - Conversion sequence not completed (or the flag event was already acknowledged and cleared by software)<br>1b - Conversion sequence completed |

| | | |
|---|---|---|
| **2**<br>**EOC** | W1C | End of Conversion Flag<br>This bit is set by hardware at the end of the conversion of a channel when a new data result is available. It is cleared by software writing 1 to it. EOC is VALID for SEQ_MOD=0-2 and can be cleared by W1C/FIFO read.<br>0b - Channel conversion not completed (or the flag event was already acknowledged and cleared by software)<br>1b - Channel conversion completed |
| **1**<br>**EOSMP** | W1C | End of Sampling Flag<br>This bit is set by hardware during the conversion, at the end of the sampling phase.<br>0b - Not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)<br>1b - End of sampling phase reached |
| **0**<br>**ADRDY** | W1C | ADC Ready<br>This bit is set by hardware once ADC is ready to accept new ADSTART command (ADC is in ready state and ADSTART=0).<br>0b - ADC not yet ready (or the flag event was already acknowledged and cleared by software)<br>1b - ADC is ready |

### 46.7.2 ADC_INT_ENABLE (Interrupt Enable Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | TRGERR_IE | 0 | FIFO_RDY_IE | ACMP_IE | 0 | | OVRIE | EOSEQIE | EOCIE | EOSMPIE | ADRDYIE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADC_INT_ENABLE field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **10**<br>**TRGERR_IE** | RW | Trigger Request Error Interrupt Enable<br>This bit is set and cleared by software to enable/disable the interrupt.<br>0b - Trigger Request Error interrupt disabled<br>1b - Trigger Request Error interrupt enabled |
| **8**<br>**FIFO_RDY_IE** | RW | FIFO Ready Interrupt Enable<br>This bit is set and cleared by software to enable/disable the FIFO ready interrupt.<br>0b - FIFO ready interrupt disabled<br>1b - FIFO ready interrupt enabled |
| **7**<br>**ACMP_IE** | RW | ADC Compare Interrupt Enable<br>This bit is set and cleared by software to enable/disable the ADC compare interrupt.<br>0b - ADC compare interrupt disabled<br>1b - ADC compare interrupt enabled |
| **4**<br>**OVRIE** | RW | Overrun Interrupt Enable<br>This bit is set and cleared by software to enable/disable the overrun interrupt.<br>0b - Overrun interrupt disabled<br>1b - Overrun interrupt enabled |
| **3**<br>**EOSEQIE** | RW | End of Conversion Sequence Interrupt Enable<br>This bit is set and cleared by software to enable/disable the end of sequence of conversions interrupt.<br>0b - EOSEQ interrupt disabled<br>1b - EOSEQ interrupt enabled |
| **2**<br>**EOCIE** | RW | End of Conversion Interrupt Enable<br>This bit is set and cleared by software to enable/disable the end of conversion interrupt.<br>0b - EOC interrupt disabled<br>1b - EOC interrupt enabled |
| **1**<br>**EOSMPIE** | RW | End of Sampling Flag Interrupt Enable<br>This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt.<br>0b - EOSMP interrupt disabled<br>1b - EOSMP interrupt enabled |
| **0**<br>**ADRDYIE** | RW | ADC Ready Interrupt Enable<br>This bit is set and cleared by software to enable/disable the ADC Ready interrupt.<br>0b - ADRDY interrupt disabled<br>1b - ADRDY interrupt enabled |

### 46.7.3 ADC_CONTROL (Control Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | ADRST | ADSTP | ADSTART | ADDIS | ADEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADC_CONTROL field descriptions**

| Field | Type | Description |
|---|---|---|
| **4**<br>**ADRST** | RW | ADC S/W reset<br>Resets all ADC logic and registers except the ADC_CONTROL register. The reset takes effect immediately and remains asserted until negated by software. After set, this bit can be cleared at any time by software.<br>0b - ADC is not reset<br>1b - ADC is reset |
| **3**<br>**ADSTP** | RW | ADC Stop Conversion Command<br>This bit is set by software to stop and discard an ongoing conversion (ADSTP Command). It is cleared by hardware when the conversion is effectively discarded.<br>0b - Write 0 has no effect. Read 0 means no ADC stop conversion command is ongoing.<br>1b - Write 1 to stop the ADC. Read 1 means that an ADSTP command is in progress. |

| | | |
|---|---|---|
| **2**<br>**ADSTART** | RW | ADC Start Conversion Command<br>This bit is set by software to start ADC conversion. Depending on the TRIGMODE configuration bits, a conversion either starts immediately (software trigger configuration) or once a hardware trigger event occurs (hardware trigger configuration).<br>It is cleared by hardware:<br>–In single conversion mode when software trigger is selected: at the assertion of the End of Conversion Sequence (EOSEQ) flag.<br>–In discontinued conversion mode when software trigger is selected: at the assertion of the End of Conversion (EOC) flag.<br>–In all cases: after the execution of the ADSTP command, at the same time as the ADSTP bit is cleared by hardware.<br>0b - Write 0 has no effect. Read 0 means no ADC conversion is ongoing.<br>1b - Write 1 to start the ADC. Read 1 means that the ADC is operating and may be converting. |
| **1**<br>**ADDIS** | RW | ADC Disable Command<br>This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state). It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).<br>0b - Write 0 has no effect. Read 0 means no ADDIS command is ongoing.<br>1b - Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress. |
| **0**<br>**ADEN** | RW | ADC Enable Command<br>This bit is set by software to enable the ADC. The ADC will be effectively ready to operate once the ADRDY flag has been set. It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.<br>0b - Write 0 has no effect. Read 0 means ADC is disabled (OFF state).<br>1b - Write 1 to enable the ADC. |

### 46.7.4 ADC_CFG1 (Configuration1 Register)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | OVRMOD | SEQGP_EN | SEQ_LEN | | | | | SEQ_MOD | | AUTO_DIS | WAIT | 0 | TRIGSRC | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | TRIGMODE | | | ALIG | RES | | 0 | | | | | | | DMAE |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | N | | | | | | | | | | N |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADC_CFG1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **30**<br>**OVRMOD** | RW | Overrun Management Mode<br>This bit is set and cleared by software and configures the way data overruns are managed.<br>0b - ADC_FIFO is preserved with the old data when an overrun is detected although adc conversion is ongoing.<br>ONLY USED for SEQ_MOD=0-2.<br>1b - ADC_FIFO is overwritten with the last conversion result when an overrun is detected. |
| **29**<br>**SEQGP_EN** | RW | Sequence Group Mode Enable<br>This bit enables or disables the sequence group mode.<br>0b - Sequence group mode is disabled, sequence mode defined in SEQ_MOD is working.<br>1b - Sequence group mode is enabled, sequence mode defined in SEQ_MOD is ignored. |
| **28-24**<br>**SEQ_LEN** | RW | Sequence Length<br>These bits define the length of sequence. The sequence length=SEQ_LEN+1. For example, SEQ_LEN=7 means sequence length is 8; SEQ_LEN=0 means sequence length is 1.<br>ONLY USED for SEQ_MOD=0-2. |
| **23-22**<br>**SEQ_MOD** | RW | Sequence Mode<br>These bits are set and cleared by software to set the ADC conversion sequence mode.<br>00b - Single sequence mode<br>01b - Continuous mode<br>10b - Dis-continuous sequence mode 0<br>11b - Dis-continuous sequence mode 1 |
| **21**<br>**AUTO_DIS** | RW | Auto Disable Mode Enable<br>0b - Auto disable mode is disabled<br>1b - Auto disable mode is enabled |
| **20**<br>**WAIT** | RW | Wait Conversion Mode<br>This bit is set and cleared by software to enable/disable wait conversion mode.<br>0b - Wait conversion mode off<br>1b - Wait conversion mode on |

| | | |
|---|---|---|
| **18-16**<br>**TRIGSRC** | RW | External Trigger Source<br>These bits are used to select which of 3 possible events can trigger conversions.<br>000b - Trigger source from PTIMER (including from TRIG_LATCH_UNIT, pre-trigger needed, ONLY USED for SEQ_MOD=3)<br>001b - Reserved<br>010b - Trigger source from TRGSEL (no pre-trigger needed, ONLY USED for SEQ_MOD=0-2)<br>011b - Trigger source from TRIG_LATCH_UNIT( pre-trigger needed, ONLY USED for SEQGP_EN=1)<br>100b-111b - Reserved |
| **13-11**<br>**TRIGMODE** | RW | Trigger Mode Select<br>These bits are used to select software trigger mode or external trigger polarity.<br>000b - Trigger detection disabled, software trigger<br>001b - Detection on rising edge<br>010b - Detection on falling edge<br>011b - Detection on both rising and falling edges<br>100b - Detection on high level voltage<br>101b - Detection on low level voltage<br>110b - Reserved<br>111b - Reserved |
| **10**<br>**ALIGN** | RW | Data Alignment<br>This bit is set and cleared by software to select right or left alignment.<br>0b - Right alignment<br>1b - Left alignment |
| **9-8**<br>**RES** | RW | Data Resolution<br>These bits are written by software to select the resolution of the conversion.<br>00b - 12 bit<br>01b - 10 bit<br>10b - 8 bit<br>11b - Reserved |
| **0**<br>**DMAEN** | RW | DMA Enable<br>This bit is set and cleared by software to enable the generation of DMA requests. This allows to use the DMA controller to manage automatically the converted data.<br>0b - DMA disabled<br>1b - DMA enabled |

### 46.7.5 ADC_CFG2 (Configuration2 Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | FWMARK | | | | | 0 | | | TRG_PRI | TRG_CLR | AVG_EN | AVG_LEN | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | CG_ACK | CG | REF_EXT | 0 | | DIV | | STCNT | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

**ADC_CFG2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **28-24**<br>**FWMARK** | RW | FIFO Watermark<br>Indicates the FIFO watermark.<br>If the number of data entries in the FIFO is greater than the FWMARK value, an ADC interrupt will be generated.<br>If the number of data entries in the FIFO is greater than the FWMARK value and bit DMAEN is set, the ADC will send a request to the DMA. |
| **20**<br>**TRG_PRI** | RW | Priority of Trigger Latch Unit<br>This bit selects the priority of the trigger request processed by trigger latch unit.<br>0b - Priority is decided on the round robin scheduling<br>1b - Priority is decided on fix priority (0>1>2>3) |
| **19**<br>**TRG_CLR** | RW | Clear Latch Trigger in Trigger Latch Unit<br>Setting this field clears all the latched triggers inside the trigger handler except the one under processing. No meaning for the read value. |
| **18**<br>**AVG_EN** | RW | Average Enable<br>Control the enable of average function.<br>0b - Average function disabled<br>1b - Average function enabled |
| **17-16**<br>**AVG_LEN** | RW | Average Number<br>Control the average number of the result.<br>00b - Average 4 conversion results<br>01b - Average 8 conversion results<br>10b - Average 16 conversion results<br>11b - Average 32 conversion results |

| | | |
|---|---|---|
| **14**<br>**CG_ACK** | RO | Ack off CG<br>0b - ADC clock source is on after clearing CG<br>1b - ADC clock source is off after setting CG |
| **13**<br>**CG** | RW | Clock Gating of ADC Clock Source<br>Before configuring DIV, must set CG and wait for CG_ACK=1.<br>After configuring DIV, must clear CG and wait for CG_ACK=0.<br>0b - Clock is on<br>1b - Clock is off |
| **12**<br>**REF_EXT** | RW | Reference SEL<br>0b - Select internal reference<br>1b - Select external reference |
| **9-8**<br>**DIV** | RW | Clock Divider Bits<br>These bits select the ADC clock divider to divide the ADC_CLK to 2^DIV.<br>Users should set these bits before ADC enable. |
| **7-0**<br>**STCNT** | RW | ADC Startup Counter Bits<br>The ADC needs a stabilization time of Tsu(~5us) before it starts converting accurately. This time is calculated by counting ADC CLK cycles until the internal counter reaches the STCNT[7:0].<br>Users should set these bits before ADC enable. |

### 46.7.6 ADC_SMPR (Sampling Rate Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SMP_OPT3 | | | | | | | | SMP_OPT2 | | | | | | | |
| RST | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SMP_OPT1 | | | | | | | | SMP_OPT0 | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**ADC_SMPR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-24**<br>**SMP_OPT3** | RW | Sampling Time Selection Option3<br>These bits are written by software to select the sampling time that applies to specific channels depended on SMPSEL bits in ADC_SC_n. The sample time is calculated as (SMP_OPT3[7:0]+2) ADC CLK cycles. The minimum value can be set is 00000010. |
| **23-16**<br>**SMP_OPT2** | RW | Sampling Time Selection Option2<br>These bits are written by software to select the sampling time that applies to specific channels depended on SMPSEL bits in ADC_SC_n. The sample time is calculated as (SMP_OPT2[7:0]+2) ADC CLK cycles. The minimum value can be set is 00000010. |
| **15-8**<br>**SMP_OPT1** | RW | Sampling Time Selection Option1<br>These bits are written by software to select the sampling time that applies to specific channels depended on SMPSEL bits in ADC_SC_n. The sample time is calculated as (SMP_OPT1[7:0]+2) ADC CLK cycles. The minimum value can be set is 00000010. |
| **7-0**<br>**SMP_OPT0** | RW | Sampling Time Selection Option0<br>These bits are written by software to select the sampling time that applies to specific channels depended on SMPSEL bits in ADC_SC_n. The sample time is calculated as (SMP_OPT0[7:0]+2) ADC CLK cycles. The minimum value can be set is 00000010. |

### 46.7.7 ADC_CMP_CTRL (Compare Control Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | ACMPEN | ACMPSGL | ACMPCH | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADC_CMP_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **7**<br>**ACMPEN** | RW | ADC Compare Enable<br>This bit is set and cleared by software.<br>0b - ADC compare disabled<br>1b - ADC compare enabled |
| **6**<br>**ACMPSGL** | RW | ADC Compare on a Single Channel or on All Channels<br>This bit is set and cleared by software to enable the ADC compare on the channel identified by the ACMPCH[5:0] bits or on all the channels.<br>0b - ADC compare enabled on all channels<br>1b - ADC compare enabled on a single channel |
| **5-0**<br>**ACMPCH** | RW | ADC Compare Channel Selection<br>These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog<br>• 000000b - ADC analog input Channel 0 monitored by ACMP<br>• 000001b - ADC analog input Channel 1 monitored by ACMP<br>• 000010b - ADC analog input Channel 2 monitored by ACMP<br>...<br>• 100010b - ADC analog input Channel 34 monitored by ACMP<br>• 100011b - ADC analog input Channel 35 monitored by ACMP<br>• 100100b - ADC analog input Channel 36 monitored by ACMP<br>• 100101b - ADC analog input Channel 37 monitored by ACMP<br>• Other values: Reserved, must not be used. |

### 46.7.8 ADC_CMP_TR (Compare Threshold Register)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | HT | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | LT | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADC_CMP_TR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **27-16**<br>**HT** | RW | ADC Compare Higher Threshold<br>These bits are written by software to define the higher threshold for the ADC compare. |
| **11-0**<br>**LT** | RW | ADC Compare Lower Threshold<br>These bits are written by software to define the lower threshold for the ADC compare. |

### 46.7.9 ADC_SGCSRn (Sequence Group Control and Status Register)

Offset : 28h+N*4h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | EOSG | 0 | | | | | | | EOSGIE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | SG_END | | | | | 0 | | | SG_START | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADC_SGCSRn field descriptions**

| Field | Type | Description |
|---|---|---|
| **24**<br>**EOSG** | RO | End of Sequence Group Flag<br>This bit is set by hardware at the end of the conversion of a sequence group. It is cleared by software writing 1 to it.<br>0b - Conversion sequence group not completed (or the flag event was already acknowledged and cleared by software)<br>1b - Conversion sequence group completed |
| **16**<br>**EOSGIE** | RW | End of Sequence Group Interrupt Enable<br>0b - EOSG interrupt disabled<br>1b - EOSG interrupt enabled |
| **12-8**<br>**SG_END** | RW | Sequence in Group End Point<br>This defines the sequence end point.<br>This value should >= SG_START, if SG_END< SG_START, ADC will use SG_START as the real end point. |
| **4-0**<br>**SG_START** | RW | Sequence in Group Start Point<br>This defines the sequence start point. |

### 46.7.10 ADC_CAL (Calibration Register)

Offset : 44h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CAL_EN | 0 | | | OFFSET | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | GAIN | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADC_CAL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**CAL_EN** | RW | Calibration Enable<br>This bit enables or disables the calibration function.<br>0b -Calibration function is disabled<br>1b - Calibration function is enabled |
| **27-16**<br>**OFFSET** | RW | Offset Value to Be Removed from the Converted Result<br>A offset defined by OFFSET[11 : 0] is subtracted from the converted results if CAL_EN is set, it is signed data where OFFSET[11] the sign. |
| **12-0**<br>**GAIN** | RW | Gain Value to be Multiplied with the Converted Result<br>A gain defined by GAIN[12 : 0]/2^12 is multiplied with the converted result if CAL_EN is set, it is unsigned data so the range is [0,2). |

### 46.7.11 ADC_FIFO_DATA (FIFO Data Register)

Offset : 4Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FIFO_DATA | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADC_FIFO_DATA field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**FIFO_DATA** | RO | ADC FIFO Data<br>This register contains the converted data when SEQ_MOD=0, 1, 2. FIFO depth is 32.<br>This data is signed if the related channel is in differential-input configuration.<br>This data is unsigned if the related channel is in single-end configuration. |

### 46.7.12 ADC_SCn (Sequence Configuration Register)

Offset : 50h+N*4h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | DIFF | SMPSEL | | COCO | AIEN | CHS | | | | | |
| W | | | | | | | | | w1c | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

**ADC_SCn field descriptions**

| Field | Type | Description |
|---|---|---|
| **10**<br>**DIFF** | RW | Differential Mode Selection<br>USED for all conversion modes.<br>If the related channel does not support differential input, configuring 1 to this bit has no effect on the channel conversion. It still converts on single input mode.<br>0b - Differential mode is disabled<br>1b - Differential mode is enabled |

| | | |
|---|---|---|
| **9-8**<br>**SMPSEL** | RW | Sample Time Options Selection<br>USED for all conversion modes.<br>00b - SMP_OPT0 is used as the sample time<br>01b - SMP_OPT1 is used as the sample time<br>10b - SMP_OPT2 is used as the sample time<br>11b - SMP_OPT3 is used as the sample time |
| **7**<br>**COCO** | W1C | Conversion Complete Flag<br>ONLY USED for SEQ_MOD=3 and SEQGP_EN=1.<br>Can be cleared by writing ADC_SC_n register or reading ADC_RESULT_n register.<br>0b - Conversion is not completed.<br>1b - Conversion is completed. |
| **6**<br>**AIEN** | RW | Interrupt Enable<br>ONLY USED for SEQ_MOD=3 and SEQGP_EN=1.<br>Enables conversion complete interrupts.<br>0b - Conversion complete interrupt is disabled.<br>1b - Conversion complete interrupt is enabled. |
| **5-0**<br>**CHS** | RW | Input Channel Select<br>USED for all conversion mode.<br>Selects one of the input channels. |

### 46.7.13 ADC_RESULTn (Result Register)

Offset : D0h+N*4h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RESULT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ADC_RESULTn field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **15-0**<br>**RESULT** | RO | ADC Results<br>These registers contain the converted data which is not saved in FIFO.<br>ONLY USED for SEQ_MOD=3 and SEQGP_EN=1.<br>This data is signed if the related channel is in differential-input configuration.<br>This data is unsigned if the related channel is in single-end configuration. |

Flagchip

# Chapter 47 Comparator (CMP)

## 47.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 47.2 Chip-specific Information

The available input and output connections for the FC7240F2M is shown below:

**Table 69. CMP connections**

| Chip | Input/Output | CMP0 | CMP1 |
|---|---|---|---|
| FC7240F2M | IN0 | CMP0_IN0 | CMP1_IN0 |
| | IN1 | CMP0_IN1 | CMP1_IN1 |
| | IN2 | CMP0_IN2 | CMP1_IN2 |
| | IN3 | CMP0_IN3 | CMP1_IN3 |
| | IN4 | CMP0_IN4 | CMP1_IN4 |
| | IN5 | CMP0_IN5 | CMP1_IN5 |
| | IN6 | CMP0_IN6 | CMP1_IN6 |
| | IN7 | CMP0_IN7 | CMP1_IN7 |
| | OUT | CMP0_OUT | CMP1_OUT |

Both CMP0 and CMP1 are available in standby mode。

For FC7240F2M, the references are connected as follows:

- vrefh0 (External Reference): VDDA (analog power supply)
- vrefh1 (Internal Reference): PMC_VREF_1P2 (1.2 V PMC bandgap reference, not available in Standby mode)

Available references for internal DAC of CMP are as follows:

**Table 70. Available references**

| Chip | CMP0 | | CMP1 | |
|---|---|---|---|---|
| FC7240F2M | vrefh0 | vrefh1 | vrefh0 | vrefh1 |
| | Yes | Yes | Yes | Yes |
| | Yes | Yes | Yes | Yes |
| | Yes | Yes | Yes | Yes |

## 47.3 Features

The Comparator (CMP) module provides a circuit for comparing two analog input voltages. It comprises a comparator, a Digital-to-Analog Converter (DAC), and an analog mux.

The DAC is an 8-bit DAC. The DAC reference can select from vrefh0 or vrefh1. See the chip-specific CMP information for the source of vrefh0 and vrefh1.

The Analog MUX provides a circuit for selecting an analog input signal from eight channels. The mux circuit is designed to operate across the full range of the supply voltage.

The features of the CMP module include:

- Two 8-to-1 channel MUXes to select input signal from eight channels
- Multiple operation modes:
  - Sampled
  - Windowed
  - Digitally Filtered

- For window and sample:
  - WINDOW/SAMPLE signal can be inverted
  - Window can be closed by CMP OUT rising, falling, or both edges
  - When a window is closed, the user can define the CMP OUT level
- Programmable hysteresis control
- Selectable inversion on comparator output
- Interrupt and DMA support
- Includes an 8-bit resolution DAC
- Selectable supply reference source for DAC
- Support channel scan in run and low power mode
- Support output bypass after analog configuration change

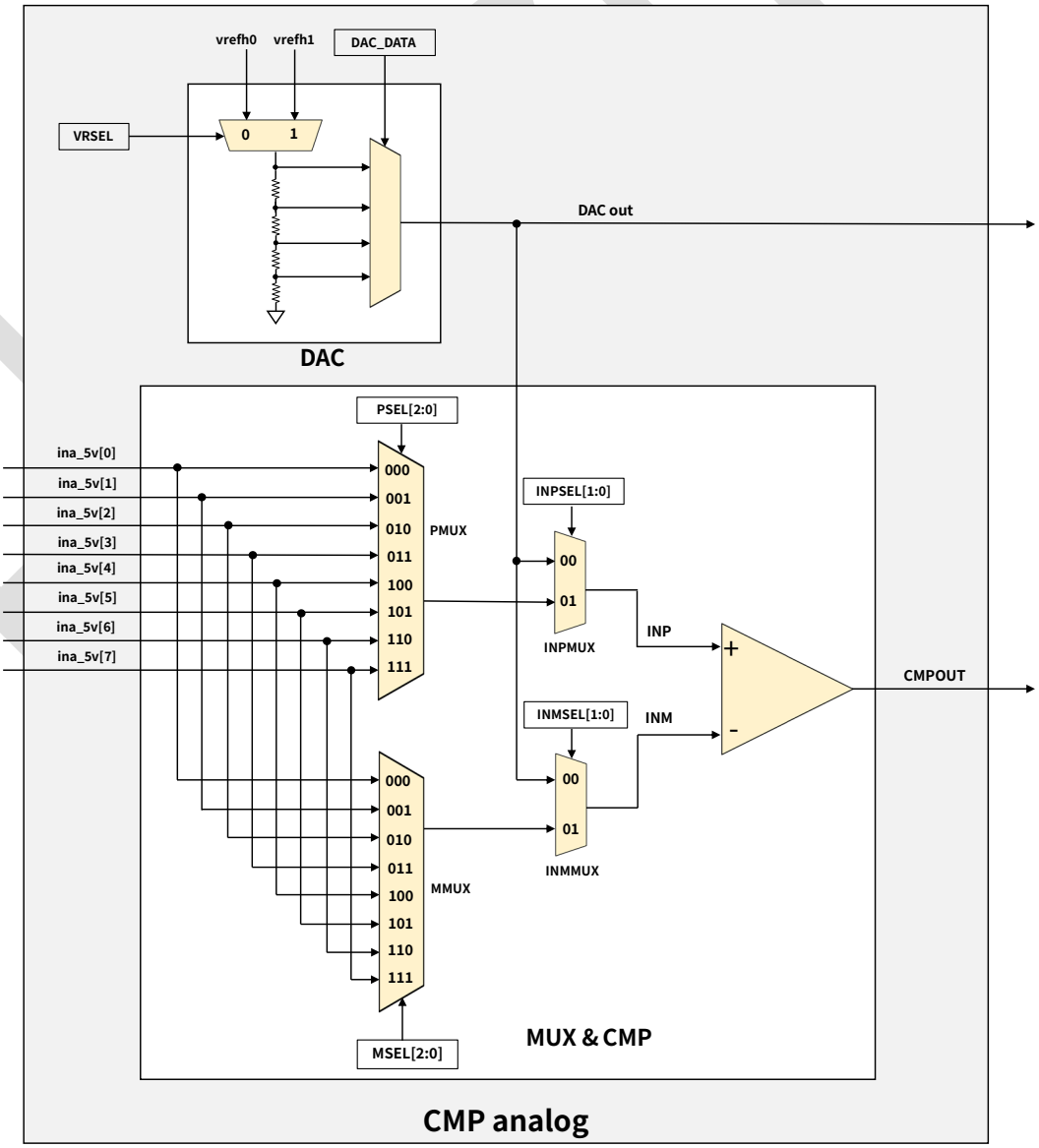
## 47.4 Overview

### 47.4.1 Instantiation Information

The FC7240F2M supports two comparators: CMP0 and CMP1. Each CMP supports an 8-bit DAC.

### 47.4.2 CMP Analog Block Diagram

**Figure 82. CMP analog block diagram**



CMP analog consists of the below blocks:

- DAC. DAC voltage output is based on 256 resistor ladders supplied by the voltage selected from vrefh0 and vrefh1. The DAC output is fed to the MUX input and can be selected as a CMP input source on either INP or INM.
- MUX. It is used to select positive and negative inputs for the analog comparator. MMUX and PMUX select inputs from 8 external sources, while INMMUX and INPMUX select inputs from the DAC output and the output from NMUX/PMUX.
- Comparator. The analog comparator outputs comparison results of positive and negative voltage levels.

### 47.4.3 DAC Reference

The 8-bit DAC sub-block supports selection of external and internal references by CMPx_DCR[VRSEL]:

- Setting CMPx_DCR[VRSEL] to 0 to select external Reference, vrefh0
- Setting CMPx_DCR[VRSEL] to 1 to select Internal Reference, vrefh1

### 47.4.4 Window Control

The CMP sample enable control signal output is from TRGSEL0. See TRGSEL0 related information for detailed information.

## 47.5 Functional Description

### 47.5.1 Function Mode

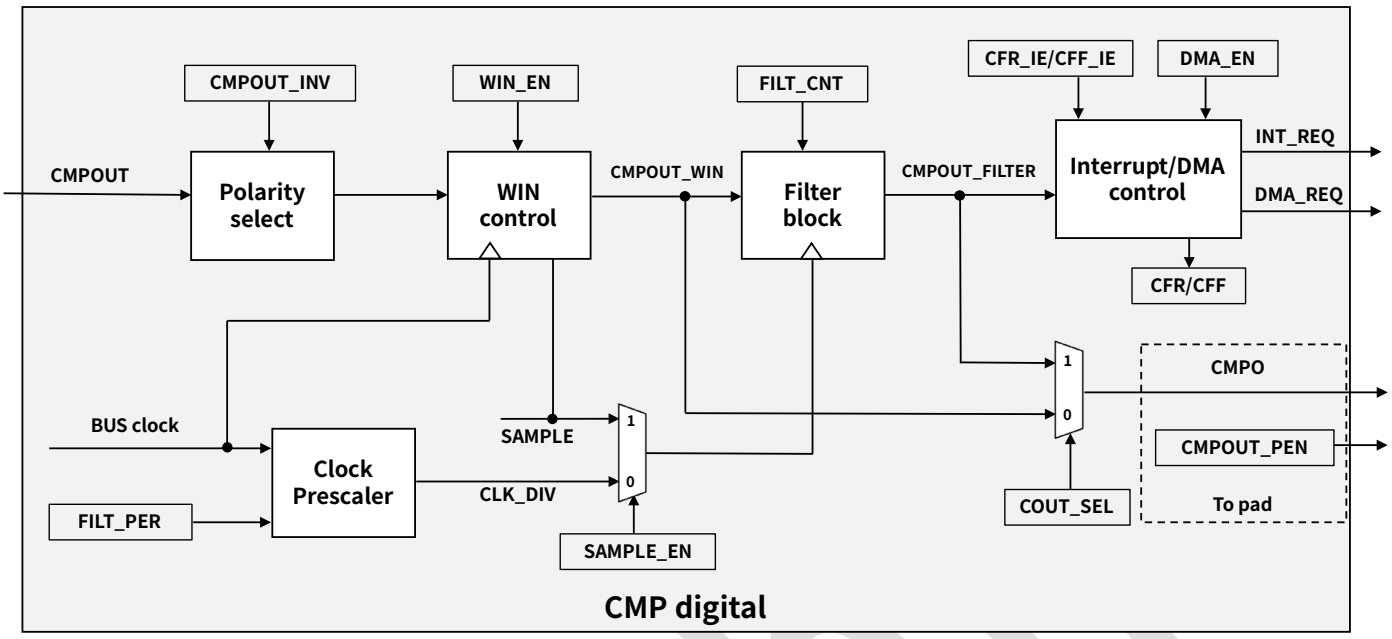
The CMP window and filter features can be combined in the following table. Individual modes are discussed in the table below.

**Table 71. CMP function mode**

| Mode | EN | WIN_EN | SAMPLE_EN | FILT_CNT | FILT_PER | Maximum Latency |
|---|---|---|---|---|---|---|
| Disable mode | 0 | X | X | X | X | N/A |
| Continuous mode 1 | 1 | 0 | X | 0x00 | X | $T_{PD}$ |
| Continuous mode 2 | 1 | 0 | 0 | X | 0x00 | $T_{PD}$ |
| Sampled, Non-filtered Mode 1 | 1 | 0 | 1 | 0x01 | X | $T_{PD} + T_{SAMPLE} + 3T_{bus}$ |
| Sampled, Non-filtered Mode 2 | 1 | 0 | 0 | 0x01 | > 0x00 | $T_{PD} + (FILT_PER * T_{bus}) + 3T_{bus}$ |
| Sampled, Filtered Mode 1 | 1 | 0 | 1 | > 0x01 | X | $T_{PD} + (FILT_CNT * T_{SAMPLE}) + 3T_{bus}$ |
| Sampled, Filtered Mode 2 | 1 | 0 | 0 | > 0x01 | > 0x00 | $T_{PD} + (FILT_CNT * FILT_PER * T_{bus}) + 3T_{bus}$ |
| Windowed Mode 1 | 1 | 1 | 0 | 0x00 | X | $T_{PD} + 2T_{bus}$ |
| Windowed Mode 2 | 1 | 1 | 0 | X | 0x00 | $T_{PD} + 2T_{bus}$ |
| Windowed/Resampled mode | 1 | 1 | 0 | 0x01 | > 0x00 | $T_{PD} + (FILT_PER * T_{bus}) + 3T_{bus}$ |
| Windowed/Filtered Mode | 1 | 1 | 0 | > 0x01 | > 0x00 | $T_{PD} + (FILT_CNT * FILT_PER * T_{bus}) + 3T_{bus}$ |

$T_{PD}$ represents the intrinsic delay of the analog component plus the polarity select logic. $T_{SAMPLE}$ is the clock period of the external sample clock. $T_{bus}$ is the period of the bus clock.

**Figure 83. CMP digital signal flow**




#### *47.5.1.1 Disabled Mode*

In disabled mode, the analog comparator is non-functional and consumes no power. CSR[CMPOUT_FILTER] and CMPO are the same as CCR1[CMPOUT_INV] in this mode.

#### *47.5.1.2 Continuous Mode*

In this mode, CMPOUT may be optionally inverted, but is not subject to external sampling or filtering. Both window control and filter blocks are completely bypassed. CSR[CMPOUT_FILTER] is updated continuously. The path from comparator input pins to output pin is operating in combinational (un-clocked) mode. CMPOUT_FILTER and CMPOUT_WIN are identical.

#### *47.5.1.3 Sampled, Non-Filtered Mode*

In this mode, the path from analog inputs to CMPOUT_WIN is combinational (un-clocked). Windowing control is completely bypassed. CMPOUT_WIN is sampled whenever a rising edge is detected on the sampling clock.

The only difference in operation between sampled, non-filtered mode 1 and sampled, non-filtered mode 2 is in how the clock to the filter block is derived. In sampled, non-filtered mode 1, the clock to filter block is externally derived while in sampled, non-filtered mode 2, the clock to filter block is internally derived.

The filter block has no other function than sample/hold of the comparator output in this mode.

#### *47.5.1.4 Sampled, Filtered Mode*

In this mode, the path from analog inputs to CMPOUT_WIN is combinational (un-clocked). Windowing control is completely bypassed. CMPOUT_WIN is sampled whenever a rising edge is detected on the sampling clock.

The only difference in operation between sampled, non-filtered mode 1 and sampled, filtered mode 1 is that CCR1[FILT_CNT]>1, which activates filter operation.

The only difference in operation between sampled, non-filtered mode 2 and sampled, filtered mode 2 is that CCR1[FILT_CNT]>1, which activates filter operation.

#### *47.5.1.5 Windowed Mode*

In this mode, CMPOUT_WIN is clocked by the bus clock whenever the window is enabled. The last latched value is held when the window is disabled. The filter block is bypassed.

If CCR1[CMPOUT_WIN_OWEN] is set, user can define CMPOUT_WIN level as CCR1[CMPOUT_WIN_OW], when window is closed, CMPOUT_WIN value is same as CCR1[CMPOUT_WIN_OW].

If CCR1[WIN_CLS] is set, the user can define the CMPO event (rising, falling, or both edges, selected by CCR1[EVT_SEL]) to close the window. The external WINDOW signal must go to zero and back to one to enable the internal window again.

If CCR1[WIN_INV] is set, the user can invert the window signal before it is used.

#### *47.5.1.6 Windowed/Resampled Mode*

This mode of operation results in an unfiltered string of comparator samples where the interval between the samples is determined by CCR1[FILT_PER] and the bus clock rate. Configuration for this mode is virtually identical to that for the windowed/filtered mode shown in the next section. The only difference is that the value of CCR1[FILT_CNT] in this mode must be 1.

#### *47.5.1.7 Windowed/Filtered Mode*

The only difference in operation between Windowed/Resampled Mode and Windowed/Filtered Mode is that CCR1[FILT_CNT]>1, which activates filter operation.

This is the most complex mode of operation for the comparator block, as it utilizes both windowing and filtering features. It also has the highest latency of any of the modes. This can be approximated: up to 2 peripheral clock synchronizations in the window function + ((CCR1[FILT_CNT] X CCR1[FILT_PER]) + 1) X peripheral clock for the filter function.

### 47.5.2 Initialization

To enable the CMP module, it is recommended to configure the control registers (CCR1, CCR2, DCR, etc.) before setting the CCR0[EN] bit. To disable the module, just clear the CCR0[EN] bit. Switch operation modes or change control register bits on the fly (when CCR0[EN] is 1) may cause some noise on the CMPOUT_FILTER or CMPOUT_WIN. So, it is recommended to disable the module before the mode switch, or any control bits change.

### 47.5.3 Enabling Filter Modes

Filter modes can be enabled by:

1) Setting CCR1[FILT_CNT] > 0x01 and
2) Setting CCR1[FILT_PER] to a nonzero value or setting CCR1[SAMPLE_EN] =1

If using the divided bus clock to drive the filter, it samples CMPOUT_WIN every CCR1[FILT_PER] bus clock cycle.

If CCR1[SAMPLE_EN] =1, the filter samples CMPOUT_WIN on each positive transition of the SAMPLE input. The output state of the filter changes when all the consecutive CCR1[FILT_CNT] samples agree that the output value has changed.

### 47.5.4 Interrupts

When the corresponding interrupt enable register bit is set, flags CSR[CFR], CSR[CFF] can generate an interrupt, assuming the DMA enable bit is not set. The interrupt is de-asserted by clearing either the flag or the interrupt enable bit.

### 47.5.5 DMA Support

When DMA support is enabled by setting CCR1[DMA_EN] and the interrupt is enabled by setting IER[CFR_IE], IER[CFF_IE], or both, the corresponding change on CMPOUT_FILTER forces a DMA transfer request rather than a CPU interrupt instead. When the DMA has completed the transfer, it sends a transfer completing indicator signal that de-asserts the DMA transfer request and clears the flags (both CSR[CFR] and CSR[CFF]) to allow a subsequent change on comparator output to occur and force another DMA request.

#### *47.5.5.1 Output Bypass*

Configurations in CCR2 and DCR supports on-the-fly change during CMP operation. The change may cause CMP output change with potential unexpected transition state, which might be unwanted for applications like peak-hold. The output of CMP analog can be bypassed before sending to digital block for polarity selection, window and sample functionality. The bypass function is enabled when CCR3[DAC_TRANS_BYP] is set to 1 and CCR3[DAC_RDY_CNT] is set to non-zero value. If there is a register write operation to CCR2 or DCR, the bypass function is triggered, and CMP analog output is latched for a period specified by CCR3[DAC_RDY_CNT]. After the period expires, the latch is removed. The clock used for output bypass is the bus clock for CMP without frequency division.

### 47.5.6 Channel Scan Mode

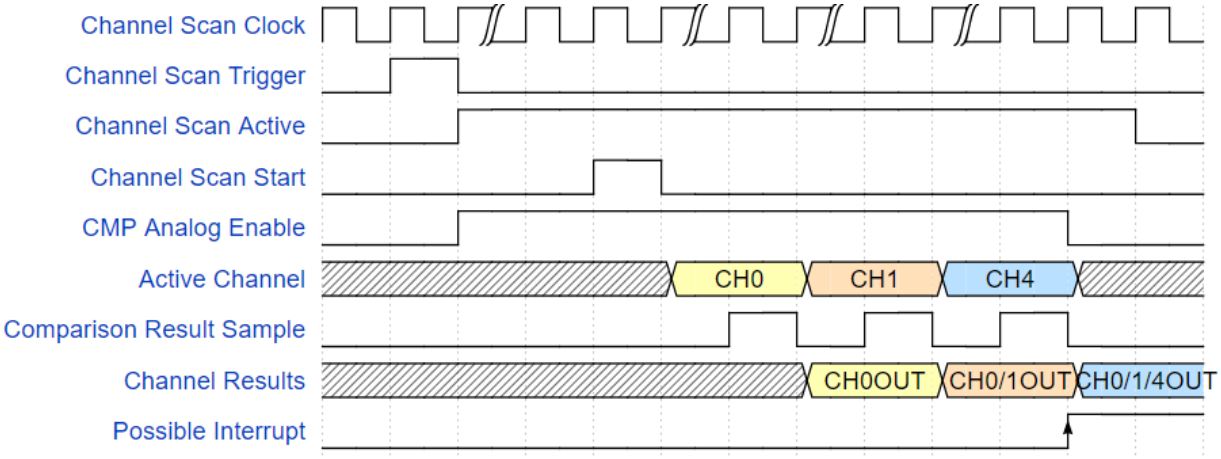
The trigger mode is enabled with both CSCR0[CS_EN] and CCR0[EN] set. After configuration, a trigger event from AONTIMER will initiate a comparison sequence from channel 0 to 7, but skipping channels that are not enabled for channel scan configured by CSCR1[CS_CHnEN], and channel specified as reference. The reference channel for either the plus side MUX or the minus side MUX is selected by CSCR1[FIXP] and CSCR1[FIXCH]. See Legal Channel Decode for valid channel configurations.

The analog circuit is enabled only during channel scan is active. It means when a trigger comes, the analog comparator is enabled and DAC analog is also enabled if used as reference input. After the current sequence completes, analog circuit is disabled automatically to save power. Channel scan mode is available in all power modes, it is suggested to configure the comparator in low speed comparison mode with CCR2[HPMD] = 0 to save power in SoC low power mode. The next trigger event should not come before the current sequence completes, which is indicated by CSSR[CS_ACTIVE].

The analog stabilization time is controlled by CSCR0[CS_INITMOD]. Make sure that the (CS_INITMOD*round robin clock period) should be longer than the initialization delay in the chip datasheet. Note that required delay for CMP low and high speed mode are different. When the stabilization process completes, the channel scan sequence begins. After the configurable number of operation clocks defined by CSCR0[CS_NSAM], the comparison result is sampled for the selected active channel.

Once all the enabled channels are compared and comparison results are sampled, if the comparison result changes from its pre-programmed state, the corresponding flag in CSSR[CS_CHnF] is set. The pre-programmed state for each channel is configured by writing to CSCSR[CS_CHnOUT] field. CSCSR[CS_CHnOUT] is updated to store the last comparison result for each channel. If any flag in CSSR[CS_CHnF] is set, flag CSR[CSF] will be set. If IER[CSF_IE] is set, an asynchronous interrupt is asserted. Note that CSR[CSF] does not support generating a DMA transfer request.

The basic flow of channel scan mode is shown in the following diagram as example. In the diagram, CSCR1[CS_CH0EN], CSCR1[CS_CH1EN], and CSCR1[CS_CH4EN] are set, so channels 0, 1, and 4 are selected for channel scan. The priority is decided by channel number. In this case, channel 0 comes first and channel 4 is the last in sequence. CSCR0[CS_NSAM] is set to 2'b01, so one clock later the comparison result of the selected channel is sampled. When channel 4 is compared, the result is sampled, and sequence ends. If any of the comparison results from channel 0, 1, or 4 changed from their programmed value (written to CSCSR[CS_CH1OUT], CSCSR[CS_CH3OUT], and CSCSR[CS_CH4OUT]), an interrupt can be generated to read flags. Software can poll the CSSR[CS_CHnF] to see which channel input(s) changed value during channel scan active.




### 47.5.7 Legal Channel Decode

CMP supports channel and DAC voltage as input sources to the comparator though MUX. The legal combinations of configurations related to source selection are listed in the below table.

| Mode | CS_EN | MSEL [2:0] | PSEL [2:0] | INMSEL [1:0] | INPSEL [1:0] | FIXP | FIXCH [2:0] | CS_CHxEN | INM | INP | Behavior |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Function** | 0 | 0-7 | X | 1 | 0 | X | X | X | Decoded from MSEL[2:0] | DAC | Channel 0-7 can be compared with DAC |
| | | X | 0-7 | 0 | 1 | X | X | X | DAC | Decoded from PSEL[2:0] | Channel 0-7 can be compared with DAC |
| | | 0-7 | 0-7 | 1 | 1 | X | X | X | Decoded from MSEL[2:0] | Decoded from PSEL[2:0] | Channel 0-7 can be compared with channel 0-7 |
| **Channel Scan** | 1 | X | X | 1 | 0 | 0 | X | 0~7 | Channel sweep in CS_CHnEN | DAC | Channel 0-7 can be swept with DAC |
| | | X | X | 0 | 1 | 1 | X | 0~7 | DAC | Channel sweep by CS_CHnEN | Channel 0-7 can be swept with DAC |
| | | X | X | 1 | 1 | 0 | 0~7 | 0~7 | Channel sweep by CS_CHnEN | Channel fixed by FIXCH[2:0] | Channel 0-7 can be swept with a fixed channel |
| | | X | X | 1 | 1 | 1 | 0~7 | 0~7 | Channel fixed by FIXCH[2:0] | Channel sweep by CS_CHnEN | Channel 0-7 can be swept with a fixed channel |

Note that in function mode, PSEL should not be equal to MSEL when both INMSEL and INPSEL is 1, or CMP will be forced off. In Channel scan mode, sweep side enabled channel should not be the same as the fixed side when INMSEL or INPSEL of the fixed side is 1, or the fixed channel in the sweep side will be skipped. “X” means all values are OK.

## 47.6 Register Definition

CMP0 base address : 4004_0000h

CMP1 base address : 4004_1000h

**Register Overview - CMP**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **8h** | CMP_CCR0 (Comparator Control Register 0) | 32 | RW | 0000_0002h | page |
| **Ch** | CMP_CCR1 (Comparator Control Register 1) | 32 | RW | 0000_0000h | page |
| **10h** | CMP_CCR2 (Comparator Control Register 2) | 32 | RW | 0000_0000h | page |
| **14h** | CMP_CCR3 (Comparator Control Register 3) | 32 | RW | 0000_0000h | page |
| **18h** | CMP_DCR (Comparator DAC Control Register) | 32 | RW | 0000_0000h | page |
| **1Ch** | CMP_IER (Comparator Interrupt Enable Register) | 32 | RW | 0000_0000h | page |
| **20h** | CMP_CSR (Comparator Status Register) | 32 | W1C | 0000_0000h | page |
| **24h** | CMP_CSCR0 (Channel Scan Control Register 0) | 32 | RW | 0000_0000h | page |
| **28h** | CMP_CSCR1 (Channel Scan Control Register 1) | 32 | RW | 0000_0000h | page |
| **2Ch** | CMP_CSCSR (Channel Scan Control and Status Register) | 32 | RW | 0000_0000h | page |
| **30h** | CMP_CSSR (Channel Scan Status Register) | 32 | RW | 0000_0000h | page |

### 47.6.1 CMP_CCR0 (Comparator Control Register 0)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | DACEN_SEL | STOP_EN | EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**CMP_CCR0 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **2**<br>**DACEN_SEL** | RW | DAC Enable Selection<br>This bit is used to select DAC enable method.<br>0b - DAC is enabled by DCR[DAC_EN]<br>1b - DAC is enabled/disabled by CCR0[EN] |
| **1**<br>**STOP_EN** | RW | STOP Mode Enable<br>This bit decides if user can enable the analog comparator or the DAC when in STOP mode.<br>0b - Comparator is disabled in STANDBY modes regardless of EN.<br>1b - Comparator is enabled in STANDBY mode if EN is active |
| **0**<br>**EN** | RW | CMP Module Enable<br>This bit enables the analog comparator module. When the module is not enabled, the analog part remains in the off state, and consumes no power.<br>0b - Analog Comparator is disabled.<br>1b - Analog Comparator is enabled. |

### 47.6.2 CMP_CCR1 (Comparator Control Register 1)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FILT_PER | | | | | | | | 0 | | | | | FILT_CNT | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | EVT_SEL | WIN_CLS | WIN_INV | CMPOUT_WIN_OW | CMPOUT_WIN_OWEN | CMPOUT_PEN | CMPOUT_SEL | CMPOUT_INV | DMA_EN | SAMPLE_EN | WIN_EN | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMP_CCR1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**FILT_PER** | RW | Filter Sample Period<br>Specifies the sampling period (in bus clock cycles) of the comparator output filter, when CCR1[SAMPLE_EN] is 0. Setting FILT_PER to 0x00 bypasses the filter when CCR1[SAMPLE_EN] is 0. Filter. This field has no effect when CCR1[SAMPLE_EN]= 1. In that case, the external SAMPLE signal is used to determine the sampling period. |

| | | |
|---|---|---|
| **18-16**<br>**FILT_CNT** | RW | Filter Sample Count<br>This field specifies the number of consecutive samples that must agree before the comparator output filter accepts a new output state.<br>000b - Filter is bypassed, CMPOUT_FILTER = CMPOUT_WIN.<br>001b - 1 consecutive sample must agree (comparator output is simply sampled).<br>010b - 2 consecutive samples must agree.<br>011b - 3 consecutive samples must agree.<br>100b - 4 consecutive samples must agree. |
| **11-10**<br>**EVT_SEL** | RW | This field is used to define which CMPO event causes window close. The event can be rising, falling or both edges.<br>00b - Rising edge causes window close.<br>01b - Falling edge causes window close.<br>10b - Both edges cause window close.<br>11b - Both edges cause window close. |
| **9**<br>**WIN_CLS** | RW | During the active window, the user can define that if CMP changes its output (CMPO), the window is closed. The user can define the CMPO event (rising, falling or both edges) to close the window. The WINDOW signal has to go to zero and back to one to enable the window again.<br>0b - WINDOW signal can not be closed by CMPO event when window mode.<br>1b - WINDOW signal can be closed by CMPO event when window mode. |
| **8**<br>**WIN_INV** | RW | This bit is used to invert the WINDOW/SAMPLE signal.<br>0b - WINDOW/SAMPLE signal is not inverted.<br>1b - WINDOW/SAMPLE signal is inverted. |
| **7**<br>**CMPOUT_WIN_OW** | RW | User can define CMPOUT_WIN level, when window is closed.<br>0b - For window mode with CMPOUT_WIN_OWEN set, once the window is closed, CMPOUT_WIN is 0.<br>1b - For window mode with CMPOUT_WIN_OWEN set, once the window is closed, CMPOUT_WIN is 1. |
| **6**<br>**CMPOUT_WIN_OWEN** | RW | User can define CMPOUT_WIN level, when window is closed.<br>0b - For window mode, once the window is closed, the CMPOUT_WIN holds the last sample value.<br>1b - For window mode, once the window is closed, user can define the CMPOUT_WIN level as CMPOUT_WIN_OW. |

| | | |
|---|---|---|
| **5**<br>**CMPOUT_PEN** | RW | Comparator Output Pin Enable<br>The CMPOUT_PEN bit enables the path from the comparator output to a selected pin.<br>0b - When CMPOUT_PEN is 0, the comparator output (after window/filter settings dependent on software configuration) is not available to a packaged pin.<br>1b - When CMPOUT_PEN is 1, and if the software has configured the comparator to own a packaged pin, the comparator output is available in a packaged pin. |
| **4**<br>**CMPOUT_SEL** | RW | Comparator Output Select<br>0b - Set CMPO to equal CMPOUT_FILTER (filtered comparator output).<br>1b - Set CMPO to equal CMPOUT_WIN (unfiltered comparator output). |
| **3**<br>**CMPOUT_INV** | RW | Comparator Invert<br>This bit allows selecting the polarity of the analog comparator function. It is also driven to the CMPO output (on both the device pin and as CSR[CMPOUT_FILTER]) when CCR0[EN] is 0. This bit is not used in trigger mode.<br>0b - Does not invert the comparator output.<br>1b - Inverts the comparator output. |
| **2**<br>**DMA_EN** | RW | DMA Enable<br>Enables the DMA transfer triggered by CMP. When this field and the corresponding interrupt enable bit are set, a DMA request is asserted when CFR or CFF is set.<br>0b - DMA is disabled.<br>1b - DMA is enabled. |
| **1**<br>**SAMPLE_EN** | RW | Sample Enable<br>If a write to this register attempts to set both SAMPLE_EN and WIN_EN, only SAMPLE_EN is set.<br>0b - Sampling mode is not selected.<br>1b - Sampling mode is selected. |
| **0**<br>**WIN_EN** | RW | Windowing Enable<br>If a write to this register attempts to set both SAMPLE_EN and WIN_EN, only SAMPLE_EN is set.<br>0b - Windowing mode is not selected.<br>1b - Windowing mode is selected. |

### 47.6.3 CMP_CCR2 (Comparator Control Register 2)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | INMSEL | | 0 | | INPSEL | | 0 | MSEL | | | 0 | PSEL | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | HYSTCTR | | 0 | | | HPMD |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMP_CCR2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **29-28**<br>**INMSEL** | RW | Selection of the input to the negative port of the comparator<br>Determines which input is selected for the minus input of the comparator.<br>00b - IN0, from the 8-bit DAC output<br>01b - IN1, from the analog 8-1 mux<br>10b - Reserved<br>11b - Reserved |
| **25-24**<br>**INPSEL** | RW | Selection of the input to the positive port of the comparator<br>Determines which input is selected for the plus input of the comparator.<br>00b - IN0, from the 8-bit DAC output<br>01b - IN1, from the analog 8-1 mux<br>10b - Reserved<br>11b - Reserved |
| **22-20**<br>**MSEL** | RW | Minus Input MUX Control<br>These bits determine which input is selected for the negative mux.<br>When an inappropriate operation selects the same input for both MUXes, the analog comparator automatically shuts down to prevent itself from becoming a noise generator, even though EN is set.<br>These bits are ignored when trigger mode is active under STOP mode.<br>000b - Input 0<br>001b - Input 1<br>...<br>111b - Input 7 |
| **18-16**<br>**PSEL** | RW | Plus Input MUX Control<br>These bits determine which input is selected for the positive mux. |

| | | |
|---|---|---|
| **5-4**<br>**HYSTCTR** | RW | Comparator hard block hysteresis control. See chip data sheet to get the actual hysteresis value with each level<br>00b - The hard block output has level 0 hysteresis internally.<br>01b - The hard block output has level 1 hysteresis internally.<br>10b - The hard block output has level 2 hysteresis internally.<br>11b - The hard block output has level 3 hysteresis internally. |
| **0**<br>**HPMD** | RW | CMP High Power Mode Select<br>0b - Low speed comparison mode is selected.<br>1b - High speed comparison mode is selected. |

### 47.6.4 CMP_CCR3 (Comparator Control Register 3)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DAC_RDY_CNT | | | | | | | | | | 0 | | | | | DAC_TRANS_BYP |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMP_CCR3 field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-6**<br>**DAC_RDY_CNT** | RW | The target counter value for bypass function<br>0000000000b - The bypass function is disabled<br>0000000001b - Target counter is 2<br>0000000010b - Target counter is 3<br>…<br>1111111111b - Target counter is 1024 |
| **0**<br>**DAC_TRANS_BYP** | RW | Analog Configuration Transition Bypass Enable<br>0b - The bypass function is disabled<br>1b - The bypass function is enabled |

### 47.6.5 CMP_DCR (Comparator DAC Control Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | DAC_DATA | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | VRSEL | 0 | | | | | | | DAC_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMP_DCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-16**<br>**DAC_DATA** | RW | DAC Output Voltage Select<br>This bit selects an output voltage from one of 256 distinct levels. DACO = (Vin/256) * (DAC_DATA + 1), so the DACO range is from Vin/256 to Vin. |
| **8**<br>**VRSEL** | RW | DAC Reference High Voltage Source Select<br>0b - vrefh0 is selected as resistor ladder network supply reference Vin.<br>1b - vrefh1 is selected as resistor ladder network supply reference Vin. |
| **0**<br>**DAC_EN** | RW | DAC Enable<br>This bit is used to enable the DAC. When the DAC is disabled, it is powered down to conserve power.<br>0b - DAC is disabled.<br>1b - DAC is enabled. |

### 47.6.6 CMP_IER (Comparator Interrupt Enable Register)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | CSF_IE | CFF_IE | CFR_IE |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMP_IER field descriptions**

| Field | Type | Description |
|---|---|---|
| **2**<br>**CSF_IE** | RW | Channel Scan Flag Interrupt Enable<br>This bit enables the interrupt when the comparison result changes for a given channel.<br>0b - The channel scan interrupt is disabled.<br>1b - The channel scan interrupt is enabled. |
| **1**<br>**CFF_IE** | RW | Comparator Flag Falling Interrupt Enable<br>When this field is set, an interrupt will be asserted when CFF is set.<br>0b - CFF interrupt is disabled.<br>1b - CFF interrupt is enabled. |
| **0**<br>**CFR_IE** | RW | Comparator Flag Rising Interrupt Enable<br>When this field is set, an interrupt will be asserted when CFR is set.<br>0b - CFR interrupt is disabled.<br>1b - CFR interrupt is enabled. |

### 47.6.7 CMP_CSR (Comparator Status Register)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | CMPOUT_FILTER | 0 | | | | | CSF | CFF | CFR |
| W | | | | | | | | | | | | | | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMP_CSR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **8**<br>**CMPOUT_FILTER** | RO | Analog Comparator Filtered Output<br>Returns the current value of the analog comparator output, when read. The field is reset to 0 and will read as CCR1[CMPOUT_INV] when the analog comparator module is disabled, that is, when CCR0[EN] =0. Writing to this field is ignored. |
| **2**<br>**CSF** | W1C | Channel Scan Flag<br>This bit is set if any channel's last comparison result is different with the pre-set value in trigger mode. This bit is cleared by writing 1 to it. CSF is cleared when CCR0[EN] or CSCR0[CS_EN] is not set.<br>0b - All channels' last comparison result with the fixed mux port are same as the pre-set value in trigger mode.<br>1b - At least one channel's last comparison result with the fixed mux port is different with the preset value in trigger mode. |
| **1**<br>**CFF** | W1C | Analog Comparator Flag Falling<br>The CFF bit is set when a falling edge on CMPOUT_FILT has been detected. CFF is cleared by writing 1 to it when CCR1[DMA_EN] is disabled. If CCR1[DMA_EN] is enabled, the flag is automatically cleared when DMA is done. CFF is cleared when CCR0[EN] is not set.<br>0b - A falling edge has not been detected on CMPOUT_FILTER.<br>1b - A falling edge on CMPOUT_FILTER has occurred. |
| **0**<br>**CFR** | W1C | Analog Comparator Flag Rising<br>The CFR bit is set when a rising edge on CMPOUT_FILTER has been detected. CFR is cleared by writing 1 to it when CCR1[DMA_EN] is disabled. If CCR1[DMA_EN] is enabled, the flag is automatically cleared when DMA is done. CFR is cleared when CCR0[EN] is not set.<br>0b - A rising edge has not been detected on CMPOUT_FILTER.<br>1b - A rising edge on CMPOUT_FILTER has occurred. |

### 47.6.8 CMP_CSCR0 (Channel Scan Control Register 0)

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | CS_INITMOD | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | CS_NSAM | | 0 | | | | | | | CS_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMP_CSCR0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **21-16**<br>**CS_INITMOD** | RW | Comparator and DAC Initialization Delay Modulus<br>The values specify the channel scan clock cycles used to determine the comparator and DAC initialization delays. Refer to datasheet for the delays.<br>For example, the initialization delay is 20us and the channel scan clock is 100KHz, then CS_INITMOD need to be set to 20us/10us = 2<br>000000b - The modulus value is 63, the same as CS_INITMOD set to 111111b<br>000000b ~ 111111b - The Initialization delay is set to CS_INITMOD * channel scan clock period |
| **9-8**<br>**CS_NSAM** | RW | Number of Sample Clocks<br>For a given channel, this field specifies how many channel scan clock cycles later the sample operation takes place.<br>00b - The comparison result is sampled as soon as the active channel is scanned in one channel scan clock.<br>01b - The sampling takes place 1 channel scan clock cycles after the next cycle of the channel scan clock.<br>10b - The sampling takes place 2 channel scan clock cycles after the next cycle of the channel scan clock.<br>11b - The sampling takes place 3 channel scan clock cycles after the next cycle of the channel scan clock |
| **0**<br>**CS_EN** | RW | Channel Scan Enable<br>0b - Channel scan operation is disabled<br>1b - Channel scan operation is enabled |

### 47.6.9 CMP_CSCR1 (Channel Scan Control Register 1)

Offset : 28h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | FIXCH | | | 0 | | | FIXP |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CS_CH7EN | CS_CH6EN | CS_CH5EN | CS_CH4EN | CS_CH3EN | CS_CH2EN | CS_CH1EN | CS_CH0EN |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMP_CSCR1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **22-20**<br>**FIXCH** | RW | Fixed Channel Selection<br>The value specifies which channel in the mux port is fixed in channel scan mode<br>000b - Channel 0 is selected as the fixed reference input for the fixed mux port<br>001b - Channel 1 is selected as the fixed reference input for the fixed mux port<br>010b - Channel 2 is selected as the fixed reference input for the fixed mux port<br>011b - Channel 3 is selected as the fixed reference input for the fixed mux port<br>100b - Channel 4 is selected as the fixed reference input for the fixed mux port<br>101b - Channel 5 is selected as the fixed reference input for the fixed mux port<br>110b - Channel 6 is selected as the fixed reference input for the fixed mux port<br>111b - Channel 7 is selected as the fixed reference input for the fixed mux port |
| **16**<br>**FIXP** | RW | Fixed Port Selection<br>The value specifies the analog mux port fixed for channel scan mode<br>0b - The Plus port is fixed. Only the inputs to the Minus port are scanned in each channel scan loop.<br>1b - The Minus port is fixed. Only the inputs to the Plus port are scanned in each channel scan loop. |
| **7**<br>**CS_CH7EN** | RW | Channel 7 Input Enable in Channel Scan Mode<br>Check chip-specific information for available channels for CMP.<br>0b - Channel 7 to the non-fixed port is disabled for scan<br>1b - Channel 7 to the non-fixed port is enabled for scan, if the same channel is selected as the reference voltage, this bit does not take effect. |
| **6**<br>**CS_CH6EN** | RW | Channel 6 Input Enable in Channel Scan Mode<br>Check chip-specific information for available channels for CMP.<br>0b - Channel 6 to the non-fixed port is disabled for scan<br>1b - Channel 6 to the non-fixed port is enabled for scan, if the same channel is selected as the reference voltage, this bit does not take effect. |

| | | |
|---|---|---|
| **5**<br>**CS_CH5EN** | RW | Channel 5 Input Enable in Channel Scan Mode<br>Check chip-specific information for available channels for CMP.<br>0b - Channel 5 to the non-fixed port is disabled for scan<br>1b - Channel 5 to the non-fixed port is enabled for scan, if the same channel is selected as the reference voltage, this bit does not take effect. |
| **4**<br>**CS_CH4EN** | RW | Channel 4 Input Enable in Channel Scan Mode<br>Check chip-specific information for available channels for CMP.<br>0b - Channel 4 to the non-fixed port is disabled for scan<br>1b - Channel 4 to the non-fixed port is enabled for scan, if the same channel is selected as the reference voltage, this bit does not take effect. |
| **3**<br>**CS_CH3EN** | RW | Channel 3 Input Enable in Channel Scan Mode<br>Check chip-specific information for available channels for CMP.<br>0b - Channel 3 to the non-fixed port is disabled for scan<br>1b - Channel 3 to the non-fixed port is enabled for scan, if the same channel is selected as the reference voltage, this bit does not take effect. |
| **2**<br>**CS_CH2EN** | RW | Channel 2 Input Enable in Channel Scan Mode<br>Check chip-specific information for available channels for CMP.<br>0b - Channel 2 to the non-fixed port is disabled for scan<br>1b - Channel 2 to the non-fixed port is enabled for scan, if the same channel is selected as the reference voltage, this bit does not take effect. |
| **1**<br>**CS_CH1EN** | RW | Channel 1 Input Enable in Channel Scan Mode<br>Check chip-specific information for available channels for CMP.<br>0b - Channel 1 to the non-fixed port is disabled for scan<br>1b - Channel 1 to the non-fixed port is enabled for scan, if the same channel is selected as the reference voltage, this bit does not take effect. |
| **0**<br>**CS_CH0EN** | RW | Channel 0 Input Enable in Channel Scan Mode<br>Check chip-specific information for available channels for CMP.<br>0b - Channel 0 to the non-fixed port is disabled for scan<br>1b - Channel 0 to the non-fixed port is enabled for scan, if the same channel is selected as the reference voltage, this bit does not take effect. |

### 47.6.10 CMP_CSCSR (Channel Scan Control and Status Register)

Offset : 2Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | CS_ACLR |
| W | | | | | | | | | | | | | | | CS_SWCLR | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CS_CH7OUT | CS_CH6OUT | CS_CH5OUT | CS_CH4OUT | CS_CH3OUT | CS_CH2OUT | CS_CH1OUT | CS_CH0OUT |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMP_CSCSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **17**<br>**CS_SWCLR** | WO | Software Clear Comparison Results<br>Writing 0 to this bit has no effect.<br>Writing 1 to this bit clears CS_CHnOUT all. Software needs to poll CS_CHnOUT to check whether they are cleared. |
| **16**<br>**CS_ACLR** | RW | Comparison Results Auto Clear Enable<br>0 - CS_CHnOUT gets overwritten by the results in next channel scan loop.<br>1 - The latest comparison results in CS_CHnOUT are automatically cleared right after next channel scan loop starts, to store new results. |
| **7**<br>**CS_CH7OUT** | RW | Comparison Result for Channel 7<br>Reading this bit returns the latest comparison result of channel 7 with the fixed mux port.<br>Writing this field defines the pre-set state of channel 7. |
| **6**<br>**CS_CH6OUT** | RW | Comparison Result for Channel 6<br>Reading this bit returns the latest comparison result of channel 6 with the fixed mux port.<br>Writing this field defines the pre-set state of channel 6. |
| **5**<br>**CS_CH5OUT** | RW | Comparison Result for Channel 5<br>Reading this bit returns the latest comparison result of channel 5 with the fixed mux port.<br>Writing this field defines the pre-set state of channel 5. |
| **4**<br>**CS_CH4OUT** | RW | Comparison Result for Channel 4<br>Reading this bit returns the latest comparison result of channel 4 with the fixed mux port.<br>Writing this field defines the pre-set state of channel 4. |
| **3**<br>**CS_CH3OUT** | RW | Comparison Result for Channel 3<br>Reading this bit returns the latest comparison result of channel 3 with the fixed mux port.<br>Writing this field defines the pre-set state of channel 3. |

| | | |
|---|---|---|
| **2**<br>**CS_CH2OUT** | RW | Comparison Result for Channel 2<br>Reading this bit returns the latest comparison result of channel 2 with the fixed mux port.<br>Writing this field defines the pre-set state of channel 2. |
| **1**<br>**CS_CH1OUT** | RW | Comparison Result for Channel 1<br>Reading this bit returns the latest comparison result of channel 1 with the fixed mux port.<br>Writing this field defines the pre-set state of channel 1. |
| **0**<br>**CS_CH0OUT** | RW | Comparison Result for Channel 0<br>Reading this bit returns the latest comparison result of channel 0 with the fixed mux port.<br>Writing this field defines the pre-set state of channel 0. |

### 47.6.11 CMP_CSSR (Channel Scan Status Register)

Offset : 30h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | CS_ACTIVE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CS_CH7F | CS_CH6F | CS_CH5F | CS_CH4F | CS_CH3F | CS_CH2F | CS_CH1F | CS_CH0F |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**CMP_CSSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **16**<br>**CS_ACTIVE** | RO | Channel Scan Active<br>This bit indicates whether channel scan is active for the moment. Writing to this bit does not take effect.<br>0b - Channel scan is inactive<br>1b - Channel scan is active |

| | | |
|---|---|---|
| **7**<br>**CS_CH7F** | RW | Comparison Results of Channel 7 Changed Flag<br>This bit is set if the channel 7's last comparison result is different with the pre-set value written to CS_CH7OUT before channel scan start.<br>This bit is cleared by writing 1 to it. |
| **6**<br>**CS_CH6F** | RW | Comparison Results of Channel 6 Changed Flag<br>This bit is set if the channel 6's last comparison result is different with the pre-set value written to CS_CH6OUT before channel scan start.<br>This bit is cleared by writing 1 to it. |
| **5**<br>**CS_CH5F** | RW | Comparison Results of Channel 5 Changed Flag<br>This bit is set if the channel 5's last comparison result is different with the pre-set value written to CS_CH5OUT before channel scan start.<br>This bit is cleared by writing 1 to it. |
| **4**<br>**CS_CH4F** | RW | Comparison Results of Channel 4 Changed Flag<br>This bit is set if the channel 4's last comparison result is different with the pre-set value written to CS_CH4OUT before channel scan start.<br>This bit is cleared by writing 1 to it. |
| **3**<br>**CS_CH3F** | RW | Comparison Results of Channel 3 Changed Flag<br>This bit is set if the channel 3's last comparison result is different with the pre-set value written to CS_CH3OUT before channel scan start.<br>This bit is cleared by writing 1 to it. |
| **2**<br>**CS_CH2F** | RW | Comparison Results of Channel 2 Changed Flag<br>This bit is set if the channel 2's last comparison result is different with the pre-set value written to CS_CH2OUT before channel scan start.<br>This bit is cleared by writing 1 to it. |
| **1**<br>**CS_CH1F** | RW | Comparison Results of Channel 1 Changed Flag<br>This bit is set if the channel 1's last comparison result is different with the pre-set value written to CS_CH1OUT before channel scan start.<br>This bit is cleared by writing 1 to it. |
| **0**<br>**CS_CH0F** | RW | Comparison Results of Channel 0 Changed flag<br>This bit is set if the channel 0's last comparison result is different with the pre-set value written to CS_CH0OUT before channel scan start.<br>This bit is cleared by writing 1 to it. |

# Chapter 48 Temperature Monitor Unit (TMU)

## 48.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • Updated the description in Section *Filter Bypass;*<br>• Updated the units for $k$ and $V_{135}$ in Section *Temperature Calculation;*<br>• Updated the description of the "TF_FILT_BYP" bit field in the register section *TMU_TF_CTRL* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 48.2 Chip-specific Information

N/A

## 48.3 Features

The Temperature Monitor Unit (TMU) module provides a circuit for monitoring temperature.

The features of the TMU module include:

- Support both voltage-based temperature sensor and flag-based temperature sensor
- Support lock and unlock with code for control registers
- For voltage-based temperature sensor:
  - Support trim for temperature calculation
- For flag-based temperature sensor:
  - Support 125°C and 150°C temperature flag
  - Support filter for temperature flag
  - Support hysteresis control for temperature comparator
  - Support both read-only status and W1C flag for temperature flag
- Interrupt support

## 48.4 Overview

### 48.4.1 Voltage-based Temperature Sensor

A voltage-based temperature sensor outputs voltage levels changing across temperature, which is provided to the ADC differential channel for conversion. The temperature is calculated based on ADC results.

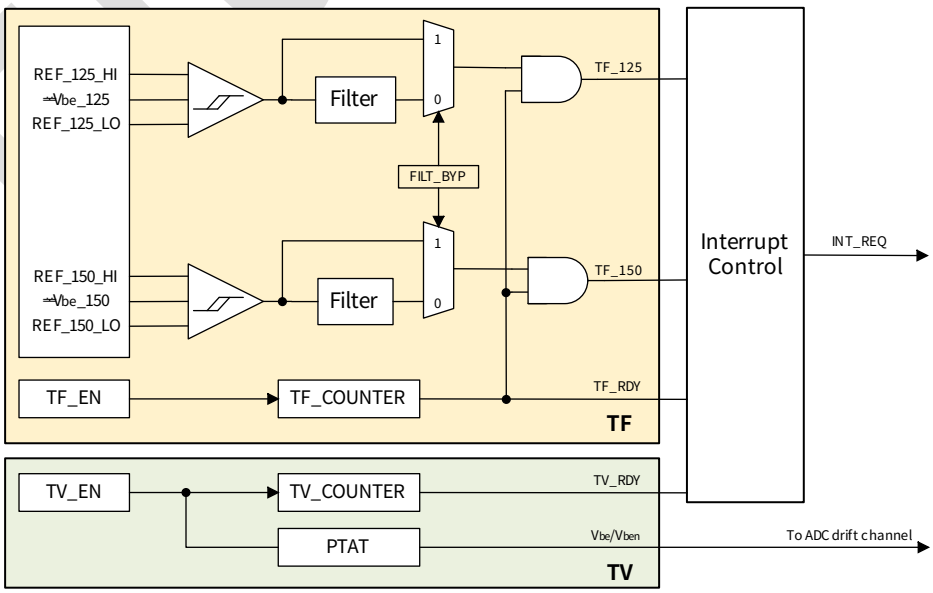
### 48.4.2 Flag-based Temperature Sensor

A flag-based temperature sensor outputs 125°C and 150°C flags directly. If the temperature crosses above 125°C /150°C, corresponding flags are set.

## 48.5 Block Diagram

The following is the block diagram of the TMU module.

**Figure 84. TMU block diagram**



## 48.6 Functional Description

### 48.6.1 Lock and Unlock

The TMU supports locking configuration registers, default in unlock state.

To Lock TV_CTRL and TF_CTRL registers, write 0 to UNLOCK[UNLOCK]. To unlock TV_CTRL and TF_CTRL, write 0xA5A50001 to UNLOCK register. Writing values other than 0xA5A50001 to UNLOCK register attempting to unlock will keep TV_CTRL and TF_CTRL registers locked without write access until the next reset.

### 48.6.2 Flag-based Temperature Sensor

#### *48.6.2.1 Initialization*

After setting TF_CTRL[TF_EN], the flag-based temperature sensor needs 11us (TYP.) to reach a stable state and get ready for temperature monitor as well as generating flags. TF_STATUS[TF_RDYF] will be set after (TF_CTRL[TF_START_CNT] *127) bus clock cycles TF_CTRL[TF_EN] is set.

If SCM_STOP_MODER0[TMU] is 1 and TF_CTRL[TF_EN] is set before entering stop mode, users need to wait for TF_STATUS[TF_RDYF] to be 1 after exiting stop mode.

#### *48.6.2.2 Interrupts*

When the corresponding interrupt enable register bit is set, flags TF_STATUS[TF_RDYF], TF_STATUS[TF_125F], TF_STATUS[TF_150F] can generate an interrupt. The interrupt is de-asserted by clearing either the flag or the interrupt enable bit.

Only interrupts from TF_STATUS[TF_125F] and TF_STATUS[TF_150F] are available in stop mode. Stop mode interrupt sources are based on real-time temperature status. For example, if the temperature reaches 125°C and SoC wakes up from stop mode by TF_STATUS[TF_125F] and SoC needs to enter stop mode again, the user needs to either disable TF_CTRL[TF_125F_IE] or cooling temperature outside below 125°C, or SoC will exit stop mode by an interrupt from TF_STATUS[TF_125F] immediately after entering stop mode.

#### *48.6.2.3 Hysteresis Control*

Hysteresis helps to eliminate glitches during temperature sensing. The flag-based temperature sensor supports 12°C hysteresis control. TF_STATUS[TF_125] and TF_STATUS[TF_150] are set when the temperature is above 125°C and 150°C.

- If TF_CTRL[TF_HYSOFF] is 0, TF_STATUS[TF_125] and TF_STATUS[TF_150] are cleared by hardware when temperature goes below 113°C and 138°C.
- If TF_CTRL[TF_HYSOFF] is 1, TF_STATUS[TF_125] and TF_STATUS[TF_150] are cleared by hardware when temperature goes below 125°C and 150°C.

#### *48.6.2.4 Filter Bypass*

Below status or flags are default filtered by a two-stage filter clocked by a functional clock, typically a clock from PCC. Refer to PCC for TMU clock information.

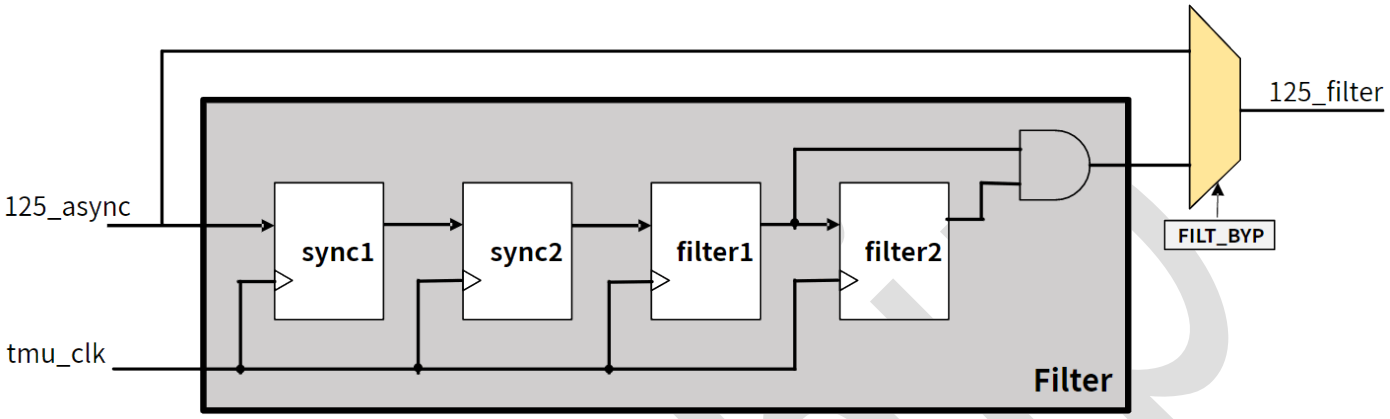
- TF_STATUS[TF_125]
- TF_STATUS[TF_150]
- TF_STATUS[TF_125F]
- TF_STATUS[TF_150F]

The filter can be bypassed by setting TF_CTRL[TF_FILT_BYP]. Hysteresis control is placed before filter logic in the circuit. It is recommended to enable hysteresis if the filter is bypassed.

If TF_125 or TF_150 is expected to generate an asynchronous interrupt to wake up from low power mode, the below configurations should be set:

1) Select 32kHz as the functional clock.
2) Enable the filter.

**Figure 85. Filter bypass function**




### 48.6.3 Voltage-based Temperature Sensor

The voltage-based temperature sensor needs cooperation with ADC in differential mode. Refer to the ADC channel section for the channel number allocated to TMU.

ADC sample time should be configured longer than the temperature sensor settling time. Refer to datasheet/specifications for settling time.

#### *48.6.3.1 Initialization*

After setting TV_CTRL[TV_EN], the voltage-based temperature sensor needs 5us (TYP.) to reach a stable state and get ready for ADC sample/conversion. TV_STATUS[TV_RDYF] will be set after (TV_CTRL[TV_START_CNT] *127) bus clock cycles TV_CTRL[TV_EN] is set.

If SCM_STOP_MODER0[TMU] is 1 and TV_CTRL[TV_EN] is set before entering stop mode, the user needs to wait for TV_STATUS[TV_RDYF] to be 1 after exiting stop mode.

#### *48.6.3.2 Interrupts*

An interrupt can be generated if TV_CTRL[TV_RDYF_IE] is 1 and TV_STATUS[TV_RDYF] is 1. The interrupt is de-asserted by clearing either the flag or the interrupt enable bit. TV_STATUS[TV_RDYF] interrupt is unavailable in stop mode.

#### *48.6.3.3 Temperature Calculation*

Temperature $T$ is calculated based on the below formula.

$$\frac{V - V_{135}}{k} + 135 = T$$

$V$ is measured voltage from ADC at temperature $T$. The unit of $k$ is $m\mathrm{V}/^{\circ}\mathrm{C}$. The unit of $V_{135}$ is $m\mathrm{V}$. The unit of $V$ is $\mathrm{mV}$. The unit of $T$ is $^{\circ}\mathrm{C}$.

The trimmed $k$ is the register value of TV_TRIM[TV_SLOPE] divided by 1000. $V_{135}$ is the measured differential results from ADC in 12bit mode in 135℃, which is stored in TV_TRIM[TV_TCODE]. Both registers are read only and used for software calculations.

## 48.7 Register Definition

TMU base address : 4004_3000h

**Register Overview - TMU**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | TMU_UNLOCK (Unlock Register) | 32 | RW | 0000_0001h | page |
| **4h** | TMU_TF_CTRL (Temperature Flag Control Register) | 32 | RW | 0000_0700h | page |
| **8h** | TMU_TF_STATUS (Temperature Flag Status Register) | 32 | RW | 0000_0000h | page |
| **Ch** | TMU_TV_CTRL (Temperature Voltage Control Register) | 32 | RW | 0000_0300h | page |
| **10h** | TMU_TV_STATUS (Temperature Voltage Status Register) | 32 | RW | 0000_0000h | page |
| **1Ch** | TMU_TV_TRIM (Temperature Voltage Trim Register) | 32 | RO | 0B05_11E4h | page |

### 48.7.1 TMU_UNLOCK (Unlock Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | UNLOCK |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**TMU_UNLOCK field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**UNLOCK** | RW | TMU Control Register Unlock<br>This bit is cleared by software to lock TF_CTRL and TV_CTRL registers. Writing 0xA5A50001 to UNLOCK register will unlock TF_CTRL and TV_CTRL. Other values except A5A50001 attempting to unlock will keep TMU lockup until the next reset.<br>0b - TF_CTRL and TV_CTRL are locked<br>1b - TF_CTRL and TV_CTRL are unlocked |

### 48.7.2 TMU_TF_CTRL (Temperature Flag Control Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | TF_150F_IE | TF_125F_IE | 0 | TF_RDYF_IE | 0 | | | | | | | TF_HYSOFF |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | TF_START_CNT | | | | 0 | | | | | | TF_FILT_BYP | TF_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TMU_TF_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **27**<br>**TF_150F_IE** | RW | Temperature over 150°C Interrupt Enable<br>An interrupt will happen if TF_150F_IE is set and TF_150F is 1.<br>TF_150F_IE is reset to 0 by hardware when TF_EN is cleared.<br>0b - Temperature over 150°C interrupt disabled<br>1b - Temperature over 150°C interrupt enabled |
| **26**<br>**TF_125F_IE** | RW | Temperature over 125°C Interrupt Enable<br>An interrupt will happen if TF_125F_IE is set and TF_125F is 1.<br>TF_125F_IE is reset to 0 by hardware when TF_EN is cleared.<br>0b - Temperature over 125°C interrupt disabled<br>1b - Temperature over 125°C interrupt enabled |
| **24**<br>**TF_RDYF_IE** | RW | Flag-based Temperature Sensor Ready Interrupt Enable<br>An interrupt will happen if TF_RDYF_IE is set and TF_RDYF is 1.<br>TF_RDYF_IE is reset to 0 by hardware when TF_EN is cleared.<br>0b - Flag-based temperature sensor ready interrupt disabled<br>1b - Flag-based temperature sensor ready interrupt enabled |
| **16**<br>**TF_HYSOFF** | RW | Flag-based Temperature Sensor Hysteresis Control<br>This bit is used to select whether hysteresis is on. The hysteresis range is 12°C. TF_HYSOFF is reset to 0 by hardware when TF_EN is cleared.<br>0b - Hysteresis is on<br>1b - Hysteresis is off |

| | | |
|---|---|---|
| **11-8**<br>**TF_START_CNT** | RW | Flag-based Temperature Sensor Startup Counter<br>Flag-based temperature sensor output needs 11us(TYP.) to reach stable state after setting CTRL[TF_EN], where flag output is valid.<br>TF_START_CNT needs to be set before CTRL[TF_EN]. STATUS[TF_RDYF] will be set after (127 * TF_START_CNT) bus clock cycles CTRL[TF_EN] is set. The minimum value allowed for TF_START_CNT is 1. |
| **1**<br>**TF_FILT_BYP** | RW | Flag-based Temperature Sensor Filter Bypass Control<br>This bit set or cleared by software to bypass or enable the digital filter for 125/150 temperature detection.<br>0b - Filter is enabled<br>1b - Filter is bypassed |
| **0**<br>**TF_EN** | RW | Flag-based Temperature Sensor Enable<br>This bit is set and cleared by software to enable/disable flag-based temperature sensor.<br>0b - Flag-based temperature is off<br>1b - Flag-based temperature is on |

### 48.7.3 TMU_TF_STATUS (Temperature Flag Status Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | TF_150 | TF_125 | 0 | | | | | | TF_150F | TF_125F | 0 | TF_RDYF |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TMU_TF_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **11**<br>**TF_150** | RO | Temperature over 150°C Status<br>TF_150 is a read-only bit indicating the current state of whether the temperature is over 150°C. This bit is only valid when CTRL[TF_EN] = 1 and flag-based temperature sensor is ready.<br>0b - Temperature is not over 150°C<br>1b - Temperature is over 150°C |
| **10**<br>**TF_125** | RO | Temperature over 125°C Status<br>TF_125 is a read-only bit indicating the current state of whether the temperature is over 125°C. This bit is only valid when CTRL[TF_EN] = 1 and flag-based temperature sensor is ready.<br>0b - Temperature is not over 125°C<br>1b - Temperature is over 125°C |
| **3**<br>**TF_150F** | RW | Temperature over 150°C Flag<br>TF_150F flag is set by hardware when the temperature exceeds 150°C. It is cleared by writing 1 to it. Users need to read TF_150 to make sure that the die temperature is below 150°C.<br>0b - Temperature has not exceeded 150°C since TF is ready or cleared by software.<br>1b - Temperature has exceeded 150°C since last time W1C. |
| **2**<br>**TF_125F** | RW | Temperature over 125°C Flag<br>TF_125F flag is set by hardware when the temperature exceeds 125°C. It is cleared by writing 1 to it. Users need to read TF_125 to make sure that the die temperature is below 125°C.<br>0b - Temperature has not exceeded 125°C since TF is ready or cleared by software.<br>1b - Temperature has exceeded 125°C since last time W1C. |
| **0**<br>**TF_RDYF** | RW | Flag-based Temperature Sensor Ready<br>TF_RDYF flag is set by hardware after TF_EN is set and the counter reaches TF_START_CNT. It is cleared by writing 1 to it.<br>0b - Flag-based temperature sensor is not ready, or cleared by software.<br>1b - Flag-based temperature sensor is ready. |

### 48.7.4 TMU_TV_CTRL (Temperature Voltage Control Register)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | TV_RDYF_IE | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | TV_START_CNT | | | 0 | | | | | | | TV_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TMU_TV_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **24**<br>**TV_RDYF_IE** | RW | Voltage-based Temperature Sensor Ready Interrupt Enable<br>An interrupt will happen if TV_RDYF is 1 and TV_RDYF_IE is set.<br>0b - Voltage-based temperature sensor ready interrupt disabled.<br>1b - Voltage-based temperature sensor ready interrupt enabled. |
| **10-8**<br>**TV_START_CNT** | RW | Voltage-based Temperature Sensor Startup Counter<br>It takes 5µs (TYP.) for the voltage-based temperature sensor output to become stable after setting CTRL[TV_EN]. After that, output is ready for ADC conversion. Besides, TV_START_CNT needs to be set before setting CTRL[TV_EN]. After CTRL[TV_EN] is set, STATUS[TV_RDYF] will be set after (127 * TV_START_CNT) bus clock cycles. The minimum value allowed for TV_START_CNT is 1. |
| **0**<br>**TV_EN** | RW | Voltage-based Temperature Sensor Enable<br>This bit is set and cleared by software to enable/disable the voltage-based temperature sensor.<br>0b - Voltage-based temperature sensor disabled.<br>1b - Voltage-based temperature sensor enabled. |

### 48.7.5 TMU_TV_STATUS (Temperature Voltage Status Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | TV_RDYF |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TMU_TV_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**TV_RDYF** | RW | Voltage-based Temperature Sensor Ready<br>TV_RDYF flag is set by hardware after TV_EN is set and the counter reaches TV_START_CNT. It is cleared by writing 1 to it.<br>0b - Voltage-based temperature sensor is not ready, or cleared by software.<br>1b - Voltage-based temperature sensor is ready. |

### 48.7.6 TMU_TV_TRIM (Temperature Voltage Trim Register)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | TV_TCODE | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | TV_SLOPE | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

**TMU_TV_TRIM field descriptions**

| Field | Type | Description |
|---|---|---|
| **27-16**<br>**TV_TCODE** | RO | Temperature Code from ADC<br>This read-only field records the ADC conversion results for voltage-based temperature sensor results in 135°C. Writing to this field has no effects. |
| **12-0**<br>**TV_SLOPE** | RO | Slope Factor<br>This read-only field records the slope factor for voltage-based temperature sensor. Writing to this field has no effects. |

# Chapter 49 Flexible Timer Unit (FTU)

## 49.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • Updated the FTU block diagram in Section *Block Diagram;*<br>• Added a note to Section *Input Capture Measurement (ICM) Mode;*<br>• Updated Figure. *CPWM0 application demo* and Figure. *CPWM1 application demo* in Section *Center-Aligned PWM (CPWM) Mode;*<br>• Updated the description in Section *Input Capture Period Measurement (ICPM) Mode;*<br>• Editorial changes |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 49.2 Chip-specific Information

The following table lists the Flexible Timer Unit (FTU) instances of each chip in the product series and summarizes features that vary across the products and instances.

| Chips | Instances | Channels | Fault Inputs | Quadrature Decoder | Modulation | Counter Width |
|---|---|---|---|---|---|---|
| FC7240F2M | FTU0 | 8 | 2 | No | No | 16-bit |
| | FTU1 | 8 | 2 | Yes | No | 24-bit |
| | FTU2 | 8 | 2 | Yes | Yes | 24-bit |
| | FTU3 | 8 | 2 | No | Yes | 16-bit |
| | FTU4 | 8 | 2 | Yes | No | 16-bit |
| | FTU5 | 8 | 2 | Yes | No | 16-bit |
| | FTU6 | 8 | 2 | No | No | 16-bit |
| | FTU7 | 8 | 2 | No | No | 16-bit |

### 49.2.1 FTU Fault Detection Inputs

For all FTU instances, fault inputs come from the TRGSEL2 trigger out. Refer to the TRGSEL2 table for details.

### 49.2.2 FTU Hardware Triggers and Synchronization

For all FTU instances, Hardware trigger 0 comes from TRGSEL0 trigger out, Hardware trigger 1 comes from TRGSEL2 trigger out, and Hardware trigger 2 comes from SCM_FTU_SYNC.

### 49.2.3 FTU Input Capture Options

The following channel 0 input capture source options are selected via SCM_FTU_ROUTING. The external pin option is selected by default.

- FTU1 channel 0 input capture = FTU1_CH0 pin or CMP0/1/2 output, controlled by SCM_FTU_ROUTING[FTU1_CH0SEL].
- FTU2 channel 0 input capture = FTU2_CH0 pin or CMP0/1/2 output, controlled by SCM_FTU_ROUTING[FTU2_CH0SEL]
- FTU2 channel 1 input capture = FTU2_CH1 pin or exclusive OR of FTU2_CH0, FTU2_CH1, and FTU2_CH2, controlled by SCM_FTU_ROUTING[FTU2_CH1SEL]
- FTU3 channel 0 input capture = FTU3_CH0 pin or exclusive OR of FTU3_CH0, FTU3_CH1, and FTU3_CH2, controlled by SCM_FTU_ROUTING[FTU3_CH0SEL]

### 49.2.4 FTU Output Modulation

The following channel output sources can be modulated by other FTU channels, and are selected via SCM_FTU_ROUTING.

- FTU0 channel pad n output = FTU0 channel n output & FTU1 channel 1 output, controlled by SCM SCM_FTU_ROUTING[FTU0_OUTSEL].
- FTU3 channel pad n output = FTU3 channel n output & FTU2 channel 1 output, controlled by SCM SCM_FTU_ROUTING[FTU3_OUTSEL].

### 49.2.5 FTU Global Time Base (GTB)

The FTU supports a global time base through the Global Time Base (GTB) feature. The global time base only allows the FTU counters to start their operation synchronously.

The Global Time Base Input (GTBIN) of each FTU instance is connected to the corresponding SCM_FTU_GTBC[FTUn_GTBC] (n: 0~7). For example, FTU0_GTBC controls FTU0 and allows software to write 1 or 0 directly.

Each SCM_FTU_GTBCM[FTUn_GTBCM] (n: 0~7) corresponds to a TSTMP1 channel trigger. If the corresponding channel trigger is active, it will load the 1 to the corresponding FTUn_GTBC when the corresponding FTUn_GTBCM mask is 1 (0 indicates no action).

For example, the desired application is:

1) Enable the FTU0/5 at time 0.
2) Enable the FTU2/3/7 at time 1.
3) Enable the FTU1/6 at time 2.
4) Enable FTU4 at time 3.

Then the configuration steps are:

1) Program the corresponding mask value.
2) Program the TSTMP1 channel 0/1/2/3 to the expected time.
3) Enable the 4 TSTMP channels at same time.

### 49.2.6 FTU Counter Width

FTU1 and FTU2 support 24-bit counter width, and other FTUs support 16-bit counter width.

## 49.3 Features

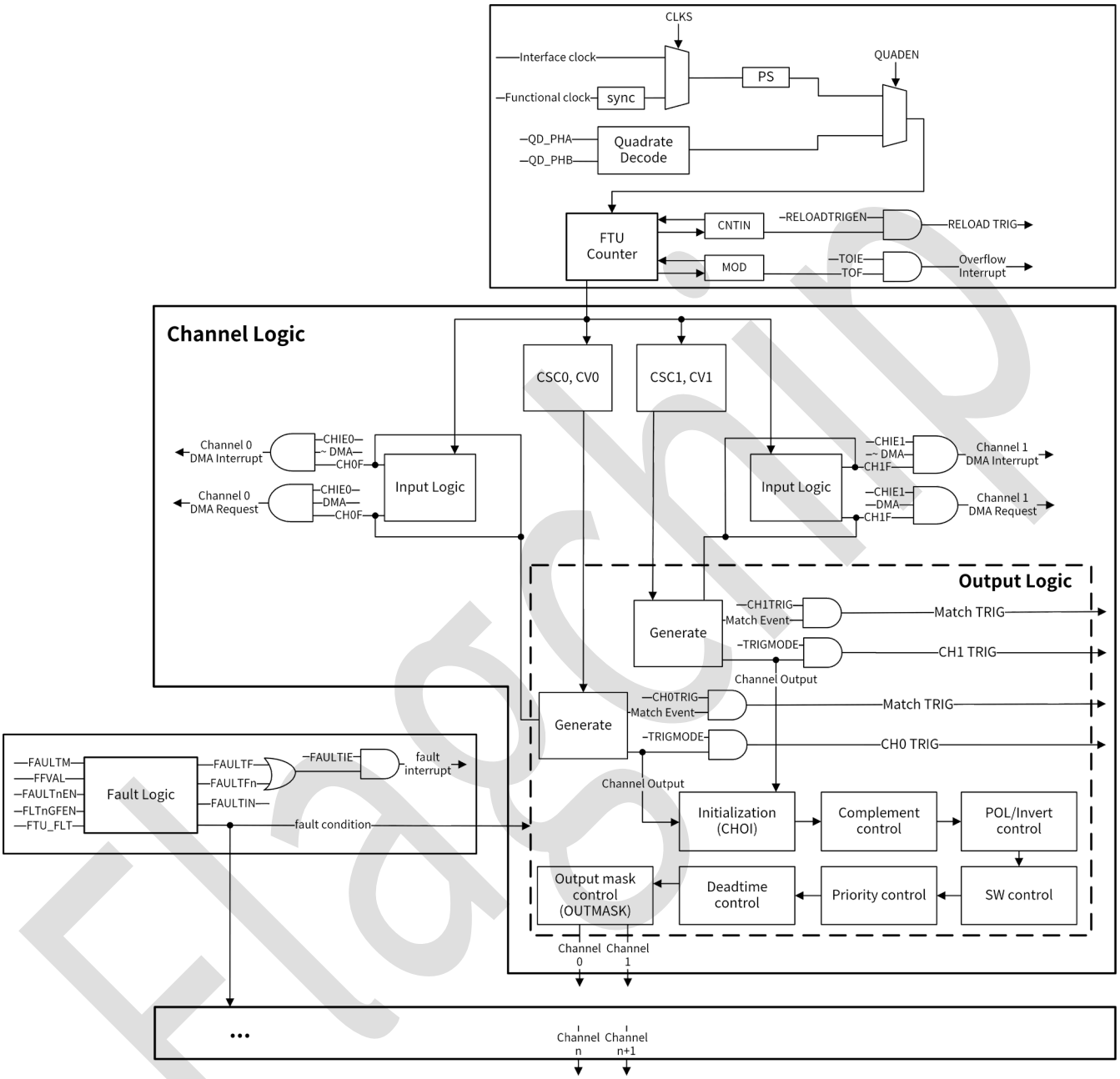
- FTU source clock is selectable
- Prescaler divided by 1, 2, 4, 8, 16, 32, 64, or 128
- 16-bit counter
- Each channel can be configured for input capture, output compare, or EPWM mode
- All channels can be configured for CPWM mode
- Deadtime supported
- Generation trigger of match point
- Software control of PWM outputs
- 2 fault inputs for global fault control
- Polarity of each channel is configurable
- Generation of an interrupt per channel
- Generation of an interrupt when the counter overflows
- Generation of an interrupt when the fault condition is detected
- Generation of an interrupt when a register reload point occurs
- Synchronized loading of write buffered FTU registers
- Register reload capacity
- Write protection for critical registers
- Direct access to input pin states
- Quadrature decoder with input glitch filters, relative position counting, and interrupt on position count or capture of position count on an external event
- FTU channels can be selected to generate a trigger pulse.

Flagchip

## 49.4 Block Diagram

The block diagram of the FTU is as follows:

**Figure 86. FTU block diagram**




## 49.5 Functional Description

### 49.5.1 Counter Mode and Events

The FTU has a 16-bit counter, and the counter clock is the selected clock divided by the prescaler. The FTU has 2 possible clock sources: the interface clock and the functional clock. The SC[CLKS] is used to select one of these two possible clock sources for the FTU counter.

The FTU input clock is mapped to the bus clock on SoC, the external pin input clock frequency must not exceed 1/4 of the FTU input clock frequency.

The FTU counter has these modes of operation:

- Up counting

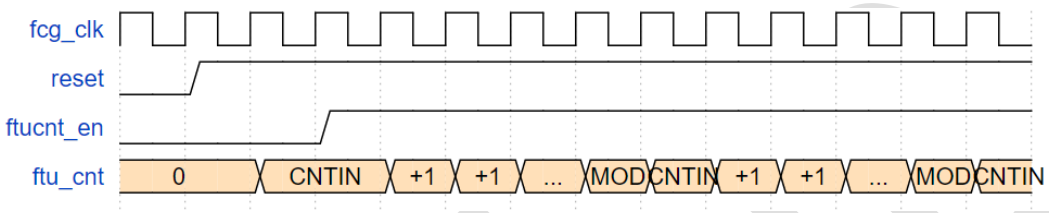
- Up-down counting
- Quadrature Decoder Mode

#### 49.5.1.1 Up Counting

Up counting is selected when: QUADEN = 0, and CPWMS = 0.

After a valid clock source is selected, the value of CNTIN is loaded into the FTU counter, and the counter increments until the value of MOD is reached, at which point the counter is reloaded with the value of CNTIN. It is recommended that the value of CNTIN be smaller than the value of MOD.

**Figure 87. FTU counter up counting**

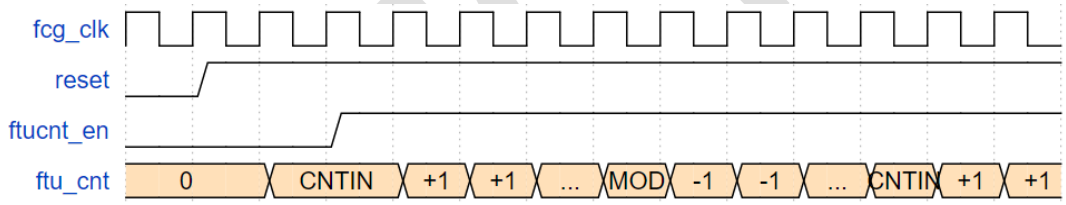



#### 49.5.1.2 Up-down Counting

Up-down counting is selected when: QUADEN = 0, and CPWMS = 1.

The counter increments until the value of MOD is reached after a valid clock source is selected, at which point the counter is decremented until it returns to the value of CNTIN and the up-down counting restarts.

**Figure 88. FTU counter up-down counting**

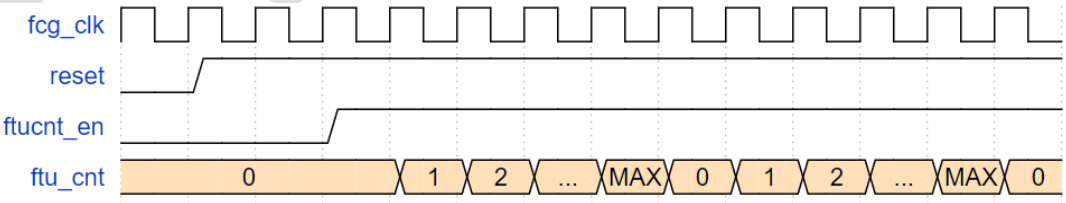



#### 49.5.1.3 Free Running Counter

If (FTUEN = 0) and (MOD = 0x0000 or MOD = 0xFFFF), the FTU counter is a free-running counter. In this case, the FTU counter runs free from 0x0000 through 0xFFFF.

If (FTUEN = 1), the counter is free running when (QUADEN =0) & (CPWMS = 0) & (CNTIN =0 and MOD = 0xFFFF).

**Figure 89. FTU counter free running counting**




#### 49.5.1.4 Counter Reset

Any one of the following cases resets the FTU counter to the value in the CNTIN register and the output of the channels to its initial value, except for channels in Output Compare mode.

- Any write to CNT.
- FTU counter synchronization.
- Input capture with ICRST = 1.

### 49.5.2 Channel Operation Mode

All following modes mentioned in this section are based on configuration with QUADEN =0, for QUADEN = 1. The FTU counter will be countered by input signals phase A/B. Refer to Quadrature Decode Mode for details.

***Note:*** The ELSB: ELSA/ MSB: MSA/ channel output/ CHF asserted, etc. are channel's related settings, for example, the configuration is channel 0, and the channel output is corresponding to channel 0 output.

| EPHASE | PHASE | CPWMS | MSB: MSA | ELSB: ELSA | ICM_MODE [2:0] | Mode | Configuration |
|---|---|---|---|---|---|---|---|
| X | X | X | XX | 00 | 000 | Pin not used for FTU | |
| 0 | 0 | 0 | 00 | Even: 01 | 001 | IC Duty-cycle Measure (ICDM) | Even CH: Capture on Rising Edge |
| | | | | Odd: 10 | | | Odd CH: Capture on Falling Edge |
| | | | | Even: 01/10 | 010 | IC Period Measure (ICPM) | Only Even CH is used to Capture Rising/Falling Edge |
| | | | | Odd: 00 | | | |
| | | | | Even: 01/10/11 | 011 | IC Edge Number Measure (ICENM) | Only Even CH is used to Capture Rising/Falling/Both Edge |
| | | | | Odd: 00 | | | |
| | | | | Even: 01/10/11 | 100 | IC Expect Edge Number Measure (ICEXPENM) | Only Even CH is used to Capture Rising/Falling/Both Edge |
| | | | | Odd: 00 | | | |
| | | | | XX | 111 | ICENM_WIND_WRITE | Config ICENM Window |
| | | | | 01 | 000 | Input Capture | Capture on Rising Edge Only |
| | | | | 10 | | | Capture on Falling Edge Only |
| | | | | 11 | | | Capture on Rising or Falling Edge |
| | | | 01 | 01 | XXX | Output Compare | Toggle Output on match |
| | | | | 10 | | | Clear Output on match |
| | | | | 11 | | | Set Output on match |
| | | | 1X | 10 | | Edge-Aligned PWM | High-true pulses (clear Output on match) |
| | | | | X1 | | | Low-true pulses (set Output on match) |
| | | 1 | XX | 10 | | Center- Aligned PWM | High-true pulses (clear Output on match-up) |
| | | | | X1 | | | Low-true pulses (set Output on match-up) |
| | 1 | 0 | XX | 10 | | Phase Shift | High-true pulses (set on channel (n) match) |
| | | | | X1 | | | Low-true pulses (clear on channel (n) match) |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | XX | 10 | | Enhanced Phase Shift | High-true pulses (set on channel (n) match) |
| | | | | X1 | | | Low-true pulses (clear on channel (n) match) |

#### *49.5.2.1 Input Capture Mode*

The Input Capture mode is selected when:

- EPHASE = 0
- PHASE = 0
- CPWMS = 0
- MSB: MSA = 0:0
- ELSB:ELSA ≠ 0:0

When a selected edge occurs on the channel input, the current value of the FTU counter is captured into the CV register, and at the same time the CHF bit is set. Writes to the CV register are ignored in input capture mode.

**Glitch Filter for Input Capture Mode**

The glitch filter (CHFVAL) is only available on channels 0, 1, 2, and 3.

The FLTPS controls the filter counter clock prescaler (available for all channels), when there is a state change in the channel input, the filter counter is reset and starts counting. The new channel input signal value is valid when the glitch filter counter reaches CHFVAL. For example, if CHFVAL is zero, the prescaler is FTU_CLK/(FLTPS+1). If CHFVAL is not zero, the precasler is FTU_CLK/(FLTPS+1)/(CHFVAL*4).

***Note:*** The value of FLTPS should be less than the value of filter/2, that is FLTPS<CHFVAL*2 (the filter value is CHFVAL*4).

#### *49.5.2.2 Input Capture Measurement (ICM) Mode*

In ICM mode, pairs of channels are used to measure the duty-cycle or period of an input signal, the number of edges during the pre-defined window, and the expected number of edges.

When an even channel is configured (such as ICM_MODE, ICM_CONT, ICDPM_START), an odd channel is automatically configured to ensure consistency in paired channels. Conversely, if only configuring an odd channel, the configuration will fail. So just configuring an even channel.

***Note:*** Make pair of channels active after all configurations and FTU clock are ready.

##### 49.5.2.2.1 Input Capture Duty-Cycle Measurement (ICDM) Mode

The ICDM mode is selected when:

- EPHASE = 0
- PHASE = 0
- CPWMS = 0
- MSB: MSA = 0:0
- ICM_MODE = 3'b001
- ELSB:ELSA = 0:1 for even channel, ELSB:ELSA = 1:0 for odd channel

In ICDM mode, the even channel is set to capture on the rising edge only, and the odd channel is set to capture on the falling edge only. After the even channel and the odd channel are configured, the pair channels are active.

It's suggested that all ICM-related configurations are done before pair channels are active.

Even channel POL is used (only used in ICDM mode) for pair channels to indicate high/low valid of an input signal. If pair channels POL=1, the odd channel will generate CHF and interrupt when the duty-cycle is measured. The value of duty-cycle is (CV1-CV0) (CV1 is CV of the odd channel, CV0 is CV of the even channel). If pair channels POL = 0, even channel will generate CHF and interrupt when the duty-cycle is measured. The value of duty-cycle is (CV0-CV1).

ICDPM_START indicates whether to take the start-point of pair channels active as an edge to measure. If ICDPM_START = 1, POL = 1 and the input signal is high when pair channels are active, the current value of the FTU counter is captured into the CV0. If ICDPM_START=1, POL=1 and the input signal is low when pair channels are active, the starting point has no effect on measurement.

ICM_CONT indicates whether to measure continuously. If ICM_CONT = 0, the measurement occurs only once. However, the measurement can be restarted by writing BIT18 of CSC0. If ICM_CONT = 1, the measurement will continue to occur.

#### 49.5.2.2.2 Input Capture Period Measurement (ICPM) Mode

The ICPM mode is selected when:

- EPHASE = 0
- PHASE = 0
- CPWMS = 0
- MSB: MSA = 0:0
- ICM_MODE = 3'b010
- ELSB:ELSA = 0:1/1:0 for even channel (odd channel is un-used, except CV1)

In ICPM mode, the even channel is set to capture on rising/falling edge. CV1 is used to store the previous value of CV0. The POL needs to be cleared due to not being used in ICPM mode.

If the even channel ELSB:ELSA = 0:1, the period (CV0-CV1) of the high valid input signal will be measured, and the CHF and an interrupt of the even channel will be generated. If the even channel ELSB:ELSA = 1:0, the period (CV0-CV1) of the low valid input signal will be measured, and the CHF and an interrupt of the even channel will be generated.

It's suggested that all ICPM-related configurations are done before pair channels are active.

ICDPM_START and ICM_CONT are similar to those in ICDM.

#### 49.5.2.2.3 Input Capture Edge Number Measurement (ICENM) Mode

The ICENM mode is selected when:

- EPHASE = 0
- PHASE = 0
- CPWMS = 0
- MSB: MSA = 0:0
- ICM_MODE = 3'b011
- ELSB:ELSA = 0:1/1:0/1:1 for even channel (odd channel is un-used, except CV1)

In ICENM mode, the rising/falling/both edge number will be measured in the pre-defined window. The pre-defined window is set in ICENM_WIND_WRITE mode (ICM_MODE=3'b111), the start-point of window is CV0, and the end-point of window is CV1. The valid window is from (CV0+1) to CV1. After the valid window, the edge number can be got by reading CSC0[31:24], and the CHF and an interrupt of even channel are generated. Writing CSC0[31:24] is useless.

It's suggested that all ICENM-related configurations are done before pair channels are active.

ICDPM_START and ICM_CONT are similar to that in ICDM.

#### 49.5.2.2.4 Input Capture Expect Edge Number Measurement (IEXPENM) Mode

The ICEXPENM mode is selected when:

- EPHASE = 0
- PHASE = 0
- CPWMS = 0
- MSB: MSA = 0:0
- ICM_MODE = 3'b100
- ELSB:ELSA = 0:1/1:0/1:1 for even channel (odd channel is un-used, except CV1 & CSC1[31:24])

The ICEXPENM mode returns the time taken to detect a desired number (ICEXP_NUM) of input edges. The time of the first edge is stored in CV0. After the time of the last edge is stored in CV1, the CHF and an interrupt of even channel is generated. The ICEXP_NUM can be set/got by writing/reading CSC1[31:24]. The internal edge counter will be cleared at (ICEXP_NUM + 1) edge when continuing to measure.

It's suggested that all ICEXPENM-related configurations are done before pair channels are active.

ICDPM_START and ICM_CONT are similar to those in ICDM.

#### *49.5.2.3 Output Compare Mode*

The Output Compare mode is selected when:

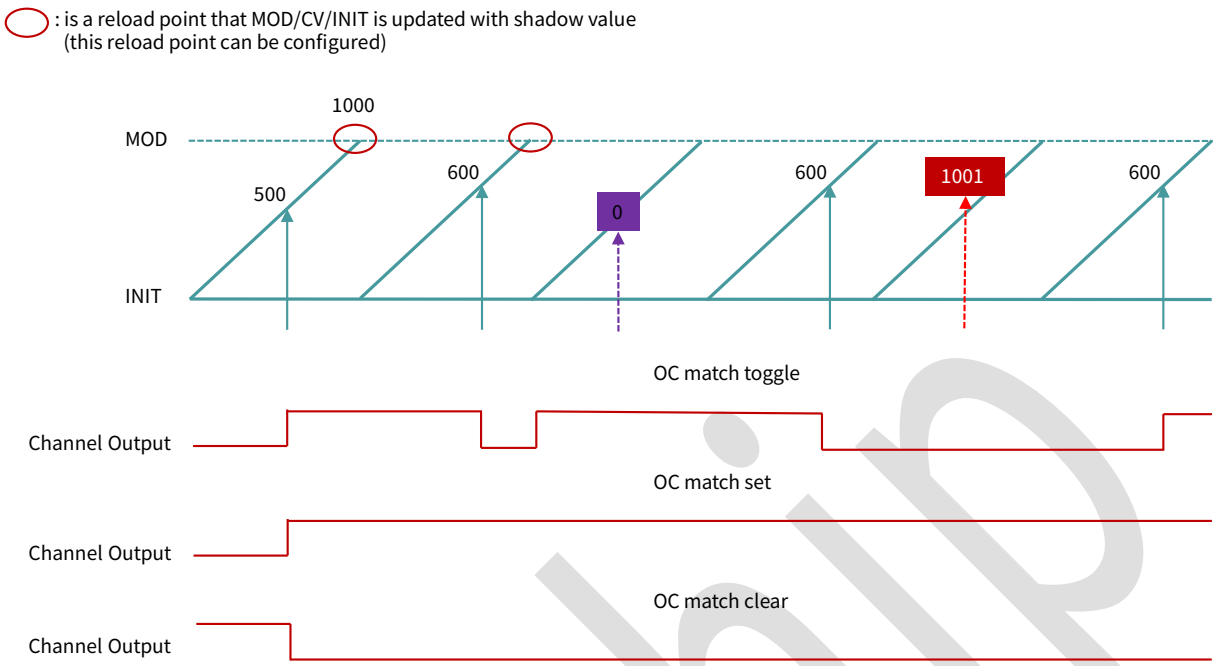
- EPHASE = 0
- PHASE = 0
- CPWMS = 0
- MSB: MSA = 0:1

If (ELSB: ELSA = 0:0) the channel output is not modified and controlled by the FTU. The CHF is set when the counter matches (counter = CV).

In Output Compare mode, when the counter matches the value in the CV register of an output compare channel, the channel output can be set, cleared, or toggled (ELSB: ELSA = 1:1/1:0/0:1). The CHF bit is set at the same time.

Flagchip

**Figure 90. Output compare application demo**




#### *49.5.2.4 Edge-Aligned PWM (EPWM) Mode*

The Edge-Aligned mode is selected when:

- EPHASE = 0
- PHASE = 0
- CPWMS = 0
- MSB = 1

If (ELSB: ELSA = 0:0), the channel output is not controlled by the FTU. The CHF is set when the counter matches.

When the counter overflows, if (ELSB: ELSA = 1:0), the channel output is forced high; If (ELSB: ELSA = X:1), the channel output is forced low. At the counter match point, if (ELSB: ELSA = 1:0), the channel output is forced low; If (ELSB: ELSA = X:1), the channel output is forced high.

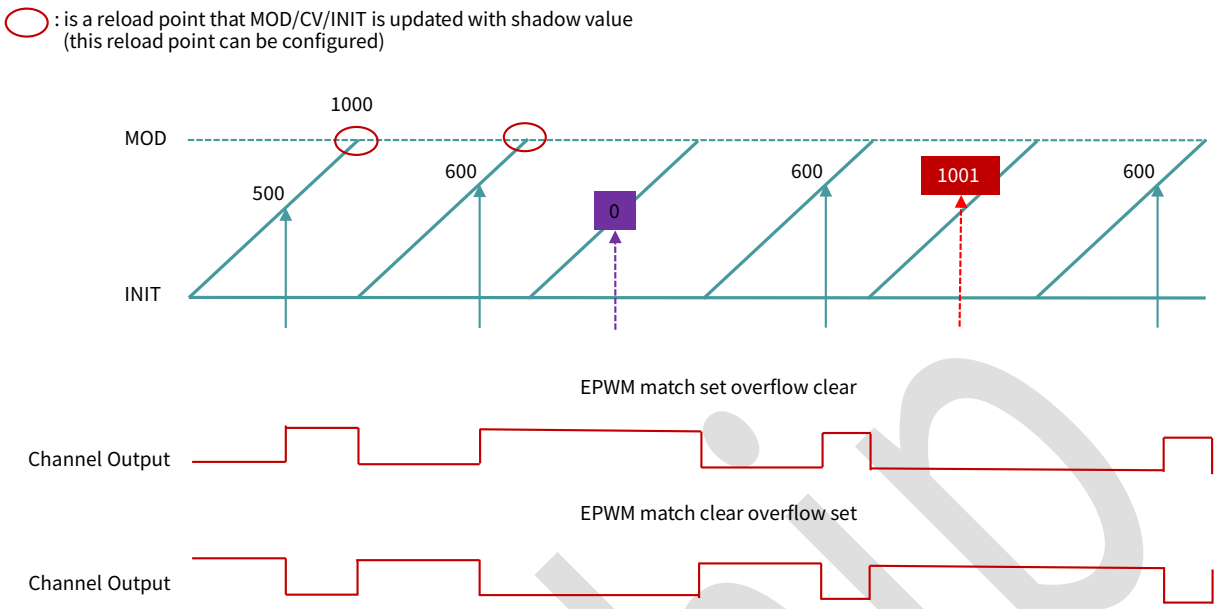
The EPWM period is determined by (MOD − CNTIN + 0x0001) and the duty cycle is determined by (CV − CNTIN).

The CHF would not be set in the following situation:

- the channel output is a 0% duty cycle EPWM signal (CV = 0x0000).
- the channel output is a 100% duty cycle EPWM signal (CV > MOD).

Flagchip

**Figure 91. EPWM application demo**




### *49.5.2.5 Center-Aligned PWM (CPWM) Mode*

The Center-Aligned mode is selected when:

- EPHASE = 0
- PHASE = 0
- CPWMS = 1

If (ELSB: ELSA = 0:0), the channel output is not controlled by the FTU. The CHF is set when counter match.

In the CPWM mode, the FTU counter counts up until it reaches MOD, then it counts down until reaches CNTIN. When a channel matches, the channel output is forced high when counting down, and it is forced low when counting up (ELSB: ELSA = 1:0). The channel output is forced low when counting down, and it is forced high when counting up (ELSB: ELSA = X:1).

The CPWM duty cycle is determined by 2 × (CV − CNTIN) and the period is determined by 2 × (MOD − CNTIN). MOD must be kept in the range of 0x0001 to 0x7FFF. Values outside this range can produce ambiguous results.

The CHF would not be set in the following situation:

- the channel output is a 0% duty cycle CPWM signal (CV = 0x0000) or (CV [15] = 1).
- the channel output is a 100% duty cycle CPWM signal (CV [15] = 0), (CV ⩾ MOD) and (MOD ≠ 0x0000).

Because there is only one counter in the FTU used by all channels, all FTU channels must be used in CPWM mode when (CPWMS = 1), and the CPWM mode must not be used when the FTU counter is a free-running counter.

**Figure 92. CPWM0 application demo**

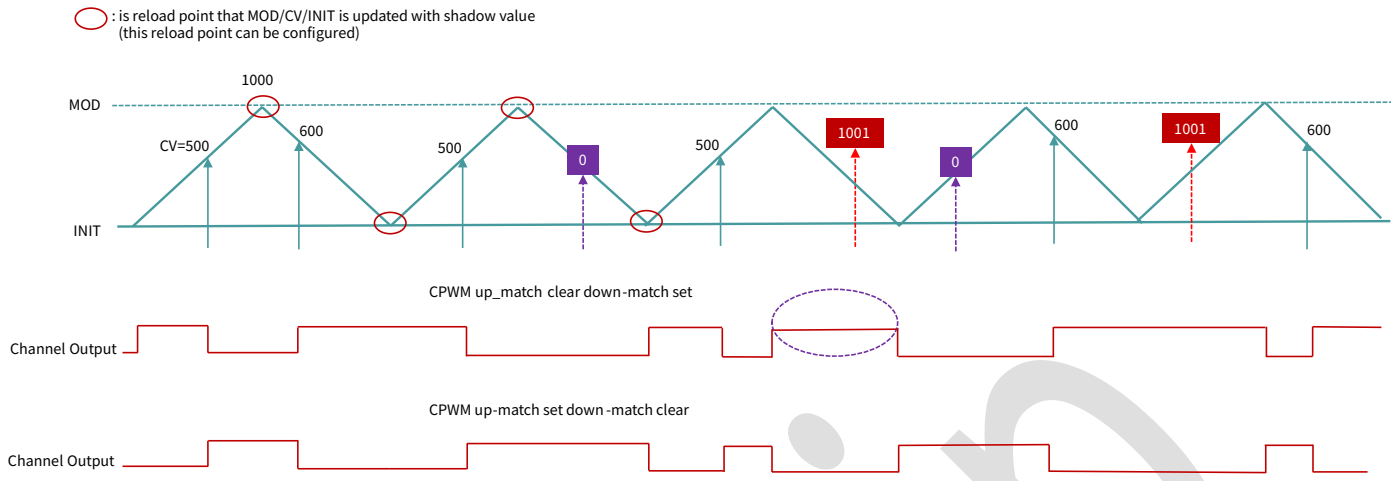
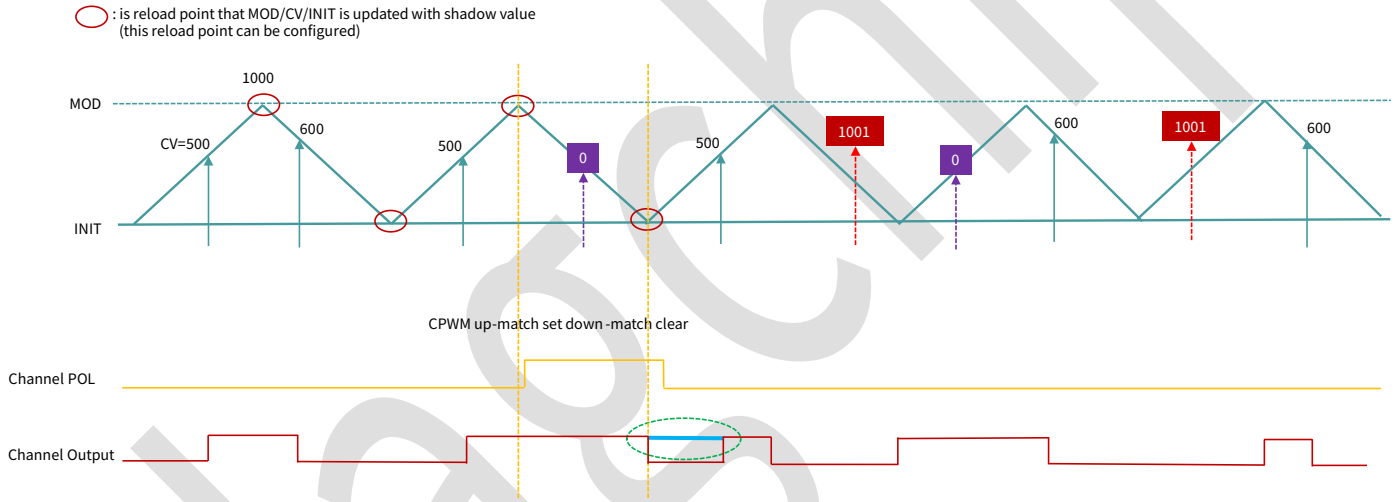



**Figure 93. CPWM1 application demo**




### *49.5.2.6 Phase Shift Mode*

The Phase Shift mode is selected when:

- EPHASE = 0
- PHASE = 1
- CPWMS = 0

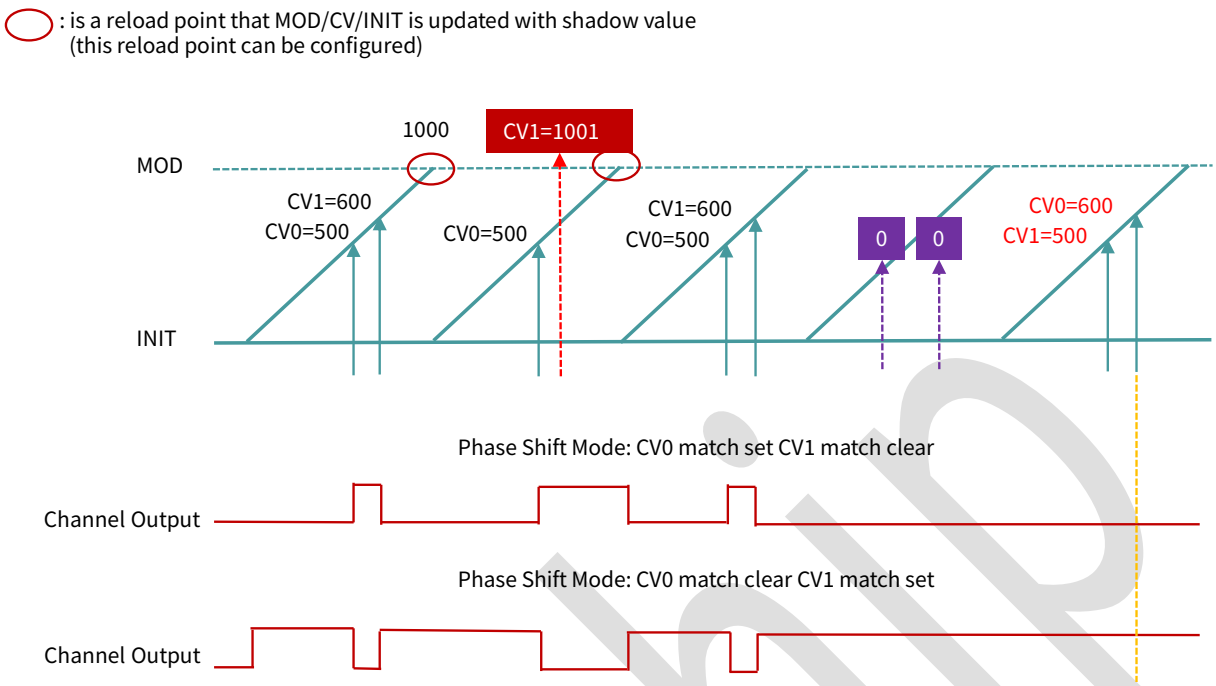
In Phase Shift mode, an even channel (n) and adjacent odd channel (n+1) are combined to generate a PWM signal in the channel (n) output. In this mode, the output controlled by channel(n) and the channel(n+1) control bit is useless.

If (ELSB: ELSA = 0:0), the channel output is not controlled by the FTU. The CHF(n) is set at the channel (n) match, the CHF(n+1) is set at channel (n+1) match.

The PWM duty cycle is determined by CV(n+1) − CV(n) and the period is determined by (MOD − CNTIN + 0x0001).

In the Phase Shift mode, when counter start, if (ELSB: ELSA = 1:0), the channel (n) output is forced low; If (ELSB: ELSA = X:1), the channel (n) output is forced high. When channel(n) match, if (ELSB: ELSA = 1:0), the channel (n) output is forced high; If (ELSB: ELSA = X:1), the channel (n) output is forced low. When channel(n+1) match, if (ELSB: ELSA = 1:0), the channel (n) output is forced low; If (ELSB: ELSA = X:1), the channel (n) output is forced high.

**Figure 94. Phase shift application demo**




### *49.5.2.7 Enhanced Phase Shift Mode*

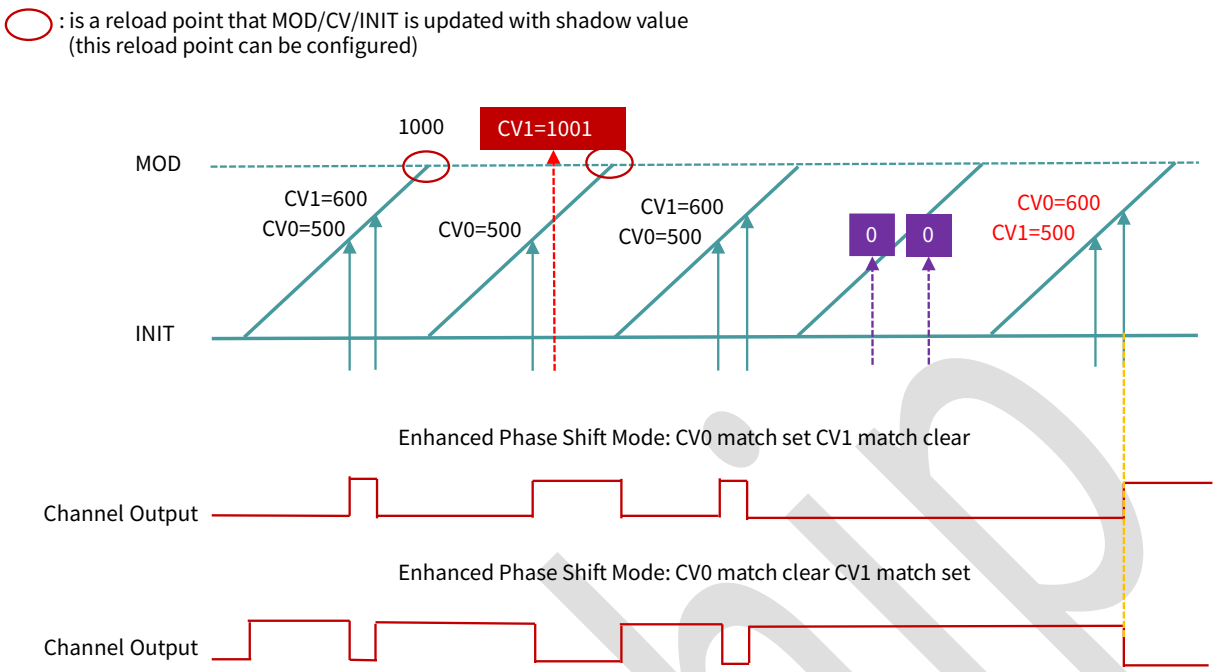
The Phase Shift mode is selected when:

- EPHASE = 1
- PHASE = 1
- CPWMS = 0

The control of the Enhanced Phase Shift mode is same as the Phase Shift mode, the difference between these two modes is that: the channel (n) match edge is fixed and the channel (n+1) match edge can be varied in Enhanced Phase Shift mode. When channel(n+1) match happened before channel(n) match in 1 PWM cycle (counter from CNTIN to MOD), that means CV of channel(n) lager than channel(n+1), in Phase Shift mode, always generate 0% or 100% duty cycle PWM in this situation, but in Enhanced Phase Shift mode, can use next cycle channel(n+1) point to generate PWM.

Flagchip

**Figure 95. Enhanced phase shift application demo**




### 49.5.3 Quadrature Decoder Mode

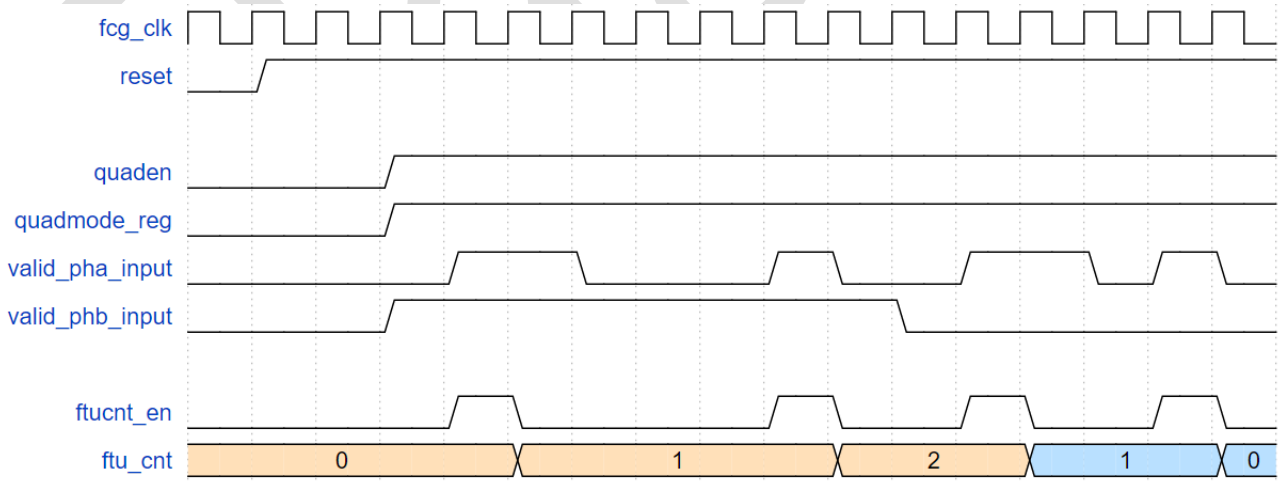
The quadrature decoder mode is enabled if QUADEN = 1, and this mode uses the input signals phase A/B to control the FTU counter.

Each one of the input signals phase A and B has a glitch filter, the phase A/B input glitch filter is enabled by PHAGFEN/ PHBGFEN, and its value is defined by CH0FVAL/ CH1FVAL. The PHAPOL/ PHBPOL selects the polarity of the phase A/B input.

The QUADMODE selects the encoding mode.

If QUADMODE = 1, the phase B value indicates the counting direction, and the phase A defines the counting rate. The FTU counter is updated when there is a rising edge at phase A input signal.

**Figure 96. Quad direction mode**




If QUADMODE = 0, the relationship between phase A and B signals indicates the counting direction, and phase A and B signals define the counting rate. The FTU counter is updated when there is a rising or falling edge either at the phase A or phase B signals.

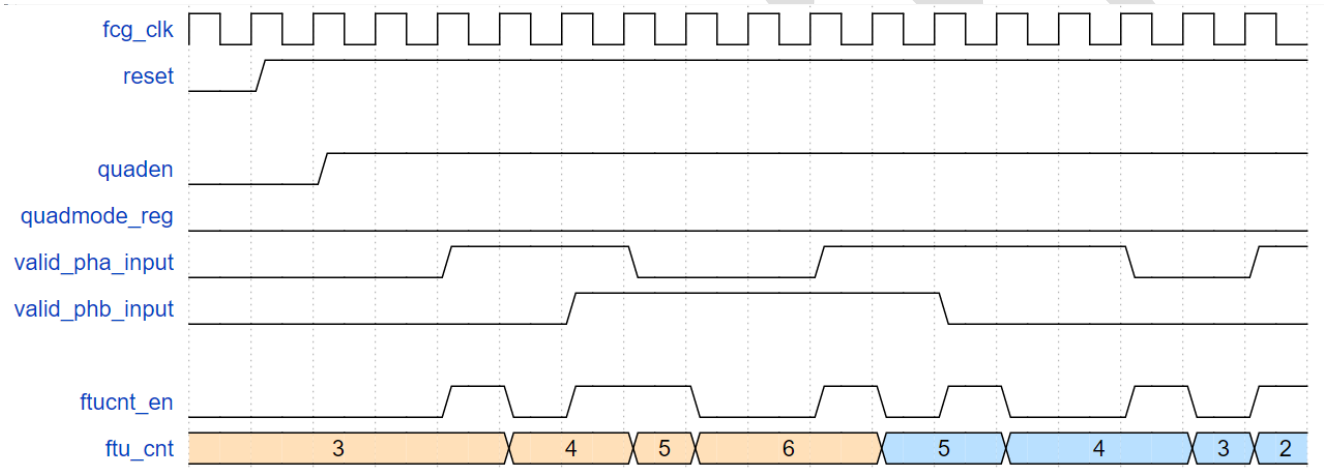
If PHAPOL = 0 and PHBPOL = 0, then the FTU counter increment happens when:

- there is a rising edge at phase A signal and phase B signal is at logic zero.
- there is a rising edge at phase B signal and phase A signal is at logic one.
- there is a falling edge at phase B signal and phase A signal is at logic zero.
- there is a falling edge at phase A signal and phase B signal is at logic one.

and the FTU counter decrement happens when:

- there is a falling edge at phase A signal and phase B signal is at logic zero.
- there is a falling edge at phase B signal and phase A signal is at logic one.
- there is a rising edge at phase B signal and phase A signal is at logic zero.
- there is a rising edge at phase A signal and phase B signal is at logic one.

**Figure 97. Quad phase mode**




### 49.5.4 Registers Updated from Write Buffers

When the FTU clock is not off, write to the below special registers will not be directly written to the real register, but be written to its register buffer, and the value in the register buffer will be loaded to the register when the update condition is established.

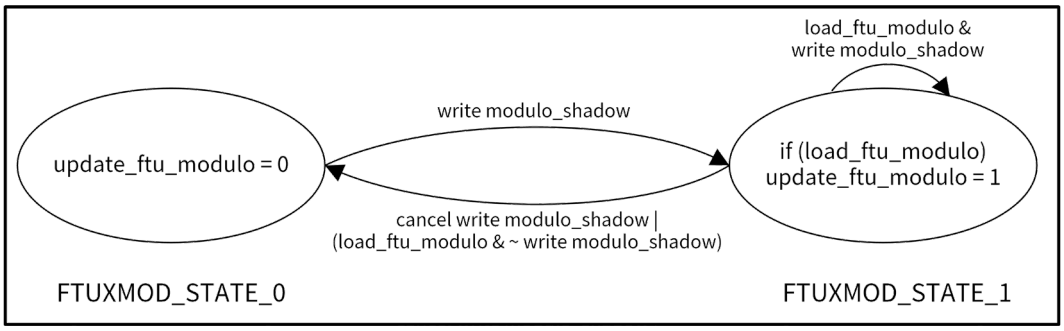
#### *49.5.4.1 MOD Register Update*

MOD register can be updated when:

- a counter overflows, that is counting from the max value to the next value (FTUEN = 0).
- the MOD register synchronization (FTUEN = 1).
- the Reload Points (LDOK = 1).

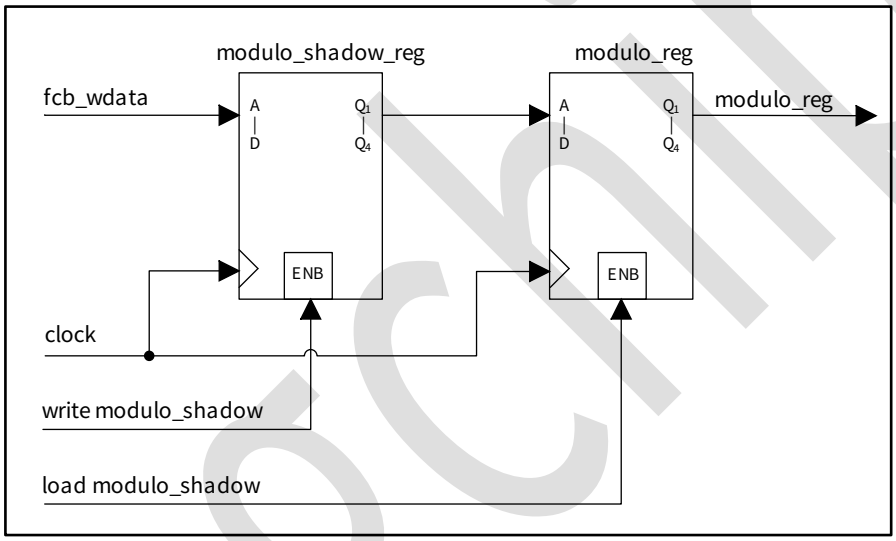
The MOD Update FSM is below. When the FSM is in state 0, the update_ftu_modulo is zero. If there is a write action to the MOD shadow register, the FSM jump to state 1, and the update_ftu_modulo is 1 if the update actions happen (load_ftu_modulo = 1).

**Figure 98. MOD update FSM**




The MOD is updated when load_ftu_modulo is high:

**Figure 99. MOD update**




#### *49.5.4.2 CNTIN Register Update*

CNTIN register can be updated when:

- the next FTU input clock after CNTIN was written (FTUEN = 0, or CNTINC = 0).
- the CNTIN register synchronization (FTUEN = 1, CNTINC = 1, and LDOK = 0).
- the Reload Points (FTUEN = 1, CNTINC = 1, and LDOK = 1).

#### *49.5.4.3 CV Register Update*

CV register can be updated when:

- the next FTU counter change at output compare mode (FTUEN = 0, or SYNCEN = 0).
- counter overflow happened at CPWM/EPWM mode (FTUEN = 0).
- the CV register synchronization (FTUEN = 1, SYNCEN = 1).
- the Reload Points (SYNCEN = 1, and LDOK = 1).

#### *49.5.4.4 INVCTRL Register Update*

INVCTRL register can be updated when each rising edge of FTU input clock.

#### *49.5.4.5 SWOCTRL Register Update*

SWOCTRL register can be updated when each rising edge of the FTU input clock.

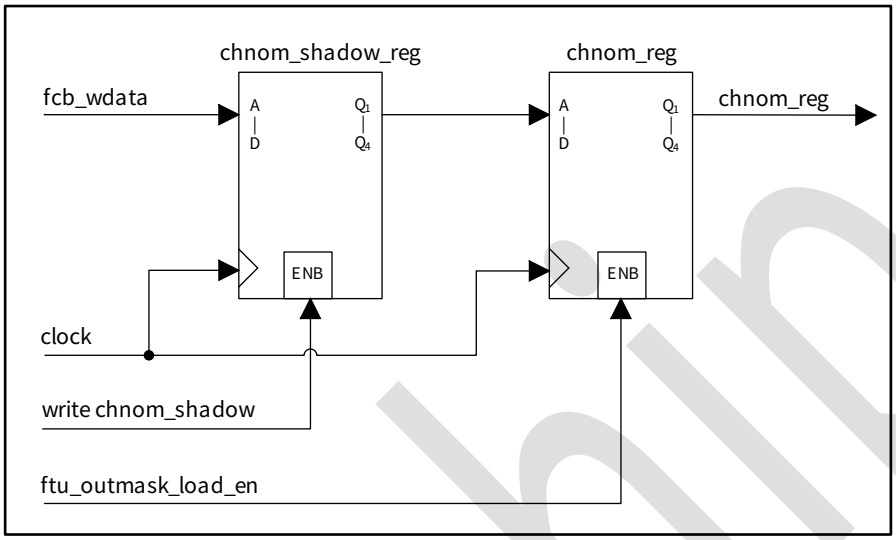
#### *49.5.4.6 OUTMASK Register Update*

The OUTMASK register can be updated when:

- each rising edge of FTU input clock (SYNCHOM = 0).
- the OUTMASK register Synchronization (SYNCHOM = 1).

The OUTMASK is updated when ftu_outmask_load_en is high:

**Figure 100. OUTMASK update**




### 49.5.5 Synchronization

#### *49.5.5.1 Synchronization Points*

The synchronization points are safe points where the registers can be updated by Synchronization, including:

- when the FTU counter overflows at up mode ((CNTMIN = 1) or (CNTMAX = 1)).
- when the FTU counter overflows at up-down mode (CNTMAX = 1).
- when the FTU counter reloads at up-down mode (CNTMIN = 1).

#### *49.5.5.2 Trigger Source*

The FTU supports multi trigger input sources to achieve synchronization, including 3 hardware triggers and 1 software trigger. For hardware trigger connection, please see Chip-specifical Information for details. If a selected trigger event happened, Synchronization by trigger started.

- Hardware triggers are enabled when TRIG(n) = 1, the hardware trigger event is when a rising edge is detected at the enabled hardware trigger, the TRIG(n) is cleared by write 0, also, it can be cleared by hardware when the corresponding hardware trigger event is detected (HWTRIGMODE = 0).
- A software trigger event occurs when 1 is written to the SWSYNC bit, the SWSYNC bit is cleared when 0 is written to it, it is also cleared by hardware when:
  - the next selected Synchronization point after that the software trigger event occurred ((PWMSYNC = 1) or (PWMSYNC = 0 and REINIT = 0)).
  - the next software trigger event occurs ((PWMSYNC = 0) and (REINIT = 1)).

#### *49.5.5.3 MOD Register Synchronization*

The MOD register synchronization is enabled if (FTUEN = 1).

MOD register synchronization is done on:

- the next Synchronization points after an enabled trigger event takes place ((PWMSYNC = 0), and (REINIT = 0)).
- the next enabled trigger event ((PWMSYNC = 0), and (REINIT = 1)).

- the next Synchronization points after the software trigger event takes place (PWMSYNC = 1).

#### *49.5.5.4 CNTIN Register Synchronization*

The CNTIN register synchronization is enabled if (FTUEN = 1, CNTINC = 1, and LDOK = 0), the same as the MOD register synchronization.

#### *49.5.5.5 CV Register Synchronization*

The CV registers synchronization is enabled if (FTUEN = 1) and (SYNCEN = 1). The synchronization mechanism is the same as the MOD register synchronization.

#### *49.5.5.6 OUTMASK Register Synchronization*

The OUTMASK register Synchronization is enabled if (SYNCHOM = 1).

OUTMASK register synchronization is done on：

- the next enabled trigger event, including software/hardware triggers (PWMSYNC = 0).
- the next enabled hardware trigger (PWMSYNC = 1).

#### *49.5.5.7 FTU Counter Synchronization*

The FTU counter synchronization can be done by Synchronization when (REINIT = 1).

FTU counter synchronization is made on:

- the next enabled trigger event, including software/hardware triggers (PWMSYNC = 0).
- the next enabled hardware trigger (PWMSYNC = 1).

### 49.5.6 Deadtime Insertion

To prevent 2 PWM output (channels (n) and (n+1)) drive the active state at the same time, the FTU supports deadtime insertion. This function enabled when:

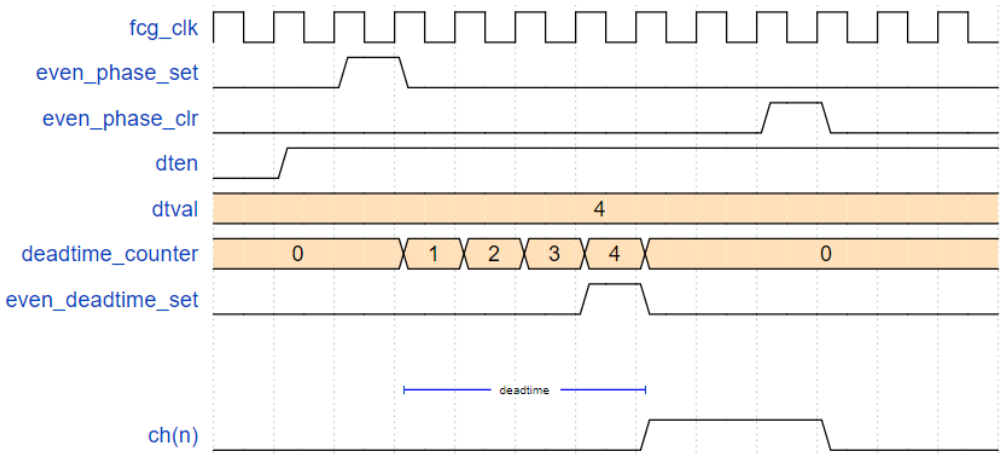
- DTEN is set
- the concatenation {DTVALEX [3:0], DTVAL [5:0]} in DEADTIME is non-zero.

The deadtime is countered by the FTU input clock (prescaler is DTPS), and the counter modulo is {DTVALEX [3:0], DTVAL [5:0]}. When the deadtime counters from 0 to its modulo, it means the end of deadtime delay, these configurations affect all FTU channels.

When the deadtime is enabled, at the channel (n) match point, the channel (n) output remains inactive until the end of the deadtime delay when the channel (n) output is active. Similarly, when the channel (n+1) matches, the channel (n+1) output remains inactive until the end of the deadtime delay when the channel (n+1) output is active.

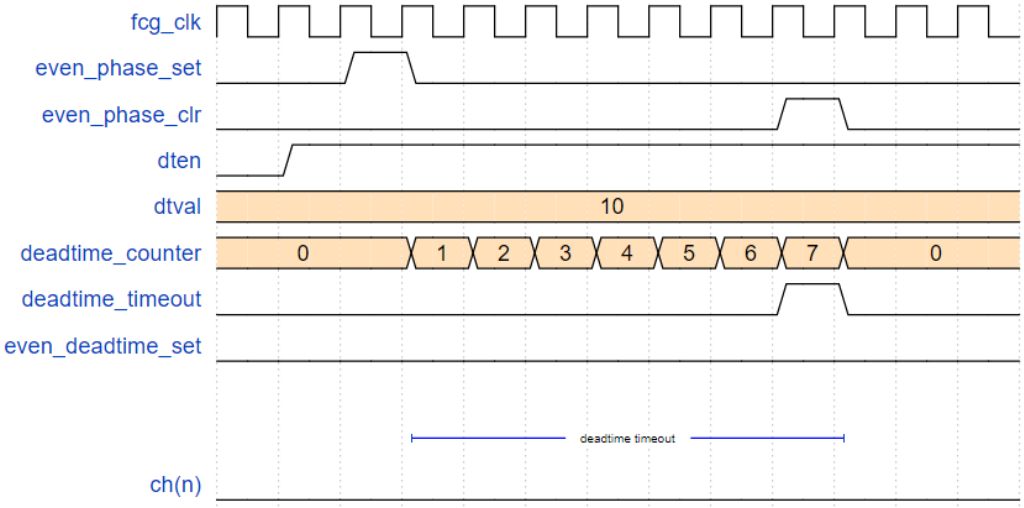
Flagchip

**Figure 101. Deadtime control**




When the deadtime insertion is too large, it will generate timeout and the channel (n) output will remain inactive:

**Figure 102. Deadtime timeout**




#### *49.5.6.1 Separated Deadtime by Pair of Channels*

The separated deadtime enabled when:

- DTEN is set
- the concatenation {DTVALEX [3:0], DTVAL [5:0]} in PAIRDEADTIME is non-zero.

The writes to the register DEADTIME also update the available registers PAIRDEADTIME. Because of this, the writes to the register DEADTIME should be done before the writes to the registers PAIRDEADTIME.

If the pair 0 of channels has a separated deadtime value, then the reads of the register DEADTIME return the read value of the register PAIRDEADTIME0. Otherwise, the read value is the value of the register DEADTIME.

### 49.5.7 Software Output Control Mode

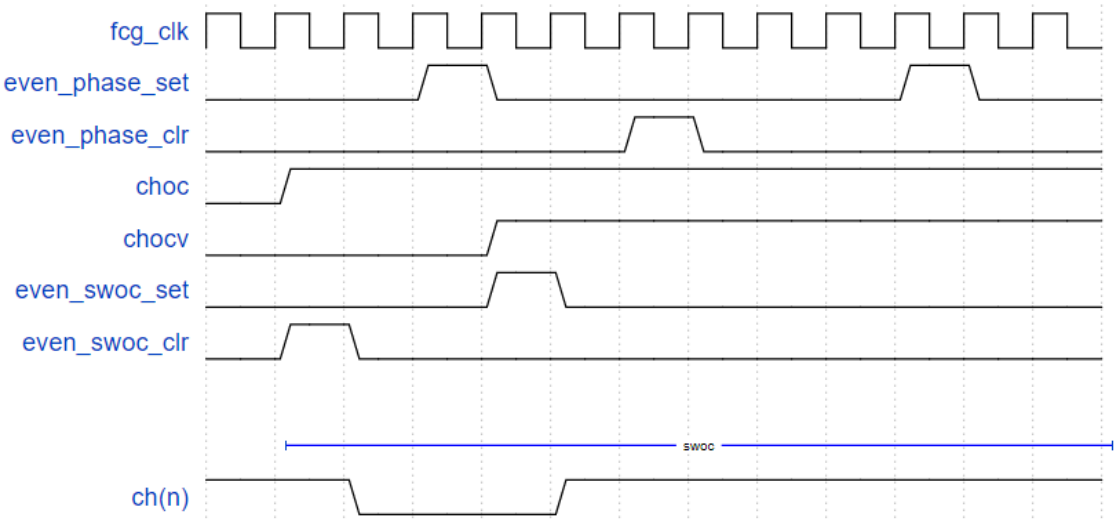
The software output control is selected when:

- QUADEN = 0
- CH(n)OC = 1

When CH(n)OC set (CH(n+1)OC should be set, too, to enable channel (n+1) software control mode), the channel output value equals to CH(n)OCV value, the channel output value equals to CH(n+1)OCV(except for when the COMP = 1 and both CH(n)OCV and CH(n+1)OCV = 1).

Channel software output control (SWOC) is controlled by register bit CH(n)OC. Below figure shows that when CH(n)OC is set, the channel output value equals to CH(n)OCV value, no matter with the initial channel set/clear (even_phase_set/ even_phase_clear).

**Figure 103. Software output control**




### 49.5.8 Fault Control

The FAULTF will set at the point which following conditions are both established:

1) The fault control is enabled (FAULTM ≠ 0:0).
2) One or more of the 4 fault inputs (maybe different numbers for some instance, please see Chip-specifical Information for detail) are enabled (FAULT(n)EN set).
3) The valid signal (after fault glitch filter, if enabled by FLT(n)GFEN, FFVAL select the value of the enabled filter, this function same as input glitch filter) drive from fault pad.

The FAULTF bit is the logic OR of FAULTF(n) bits, the flag keep set if the fault is not cleared.

The channel outputs are forced to the inactive state after fault delay (the delay time is FDLYV0 or FDLYV1 selected by FDLYSEL), or the outputs are not controlled by FTU according to the FSTATE, need (FAULTEN(n) set), the channel output is re-enabled when:

- the fault input signal (FAULTIN) returns to zero and a new PWM cycle begins (FAULTM = 1:1).
- the FAULTF bit is cleared and a new PWM cycle begins (FAULTM = 0:1 or 1:0).

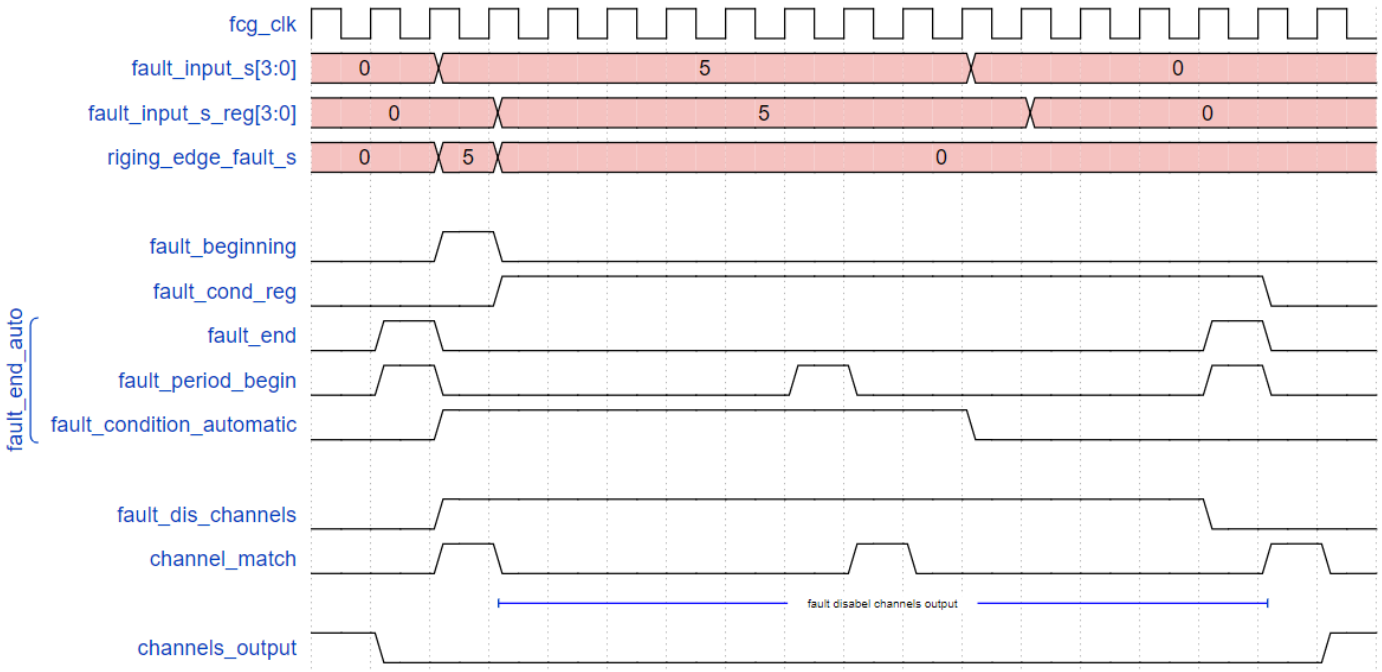
***Note:*** If the channel outputs are controlled by FTU_SWOCTRL, the channel out must be changed by writing to FTU_SWOCTRL again after the FAULTF bit is cleared and the next PWM cycle starts.

Below is the waveform of fault disabling channel output when the fault condition automatic clearing is selected (FAULTM = 2'b11). From the figure we can see that the fault condition (fault_condition_automatic) is automatically cleared with valid fault input (fault_input_s) deserting.

The fault_cond_reg is used to record the fault state. It will be set when fault rising edge is detected and will be cleared at every new PWM cycle beginning if fault condition is cleared.

The fault disabling channel output will be enable at fault beginning and during fault state, except for fault ending time. Here the channel will be asserted when channel match (The FTU counter counts to the value of CV).

**Figure 104. Fault condition automatic clearing**




Below is the waveform of fault disabling channel output when the fault condition automatic clearing is selected (FAULTM = 2'b01 or 2'b10). The clr_flag_faults indicate whether clearing the fault flags bits FAULTF(n) and the clr_flag_fault indicates whether clearing the fault flag bit FAULTF. The action of clearing fault flag will be successful only if there are no fault input. The FAULTF bit is the logic OR of FAULTF(n) bits, the flag keep set if the fault input is not cleared.

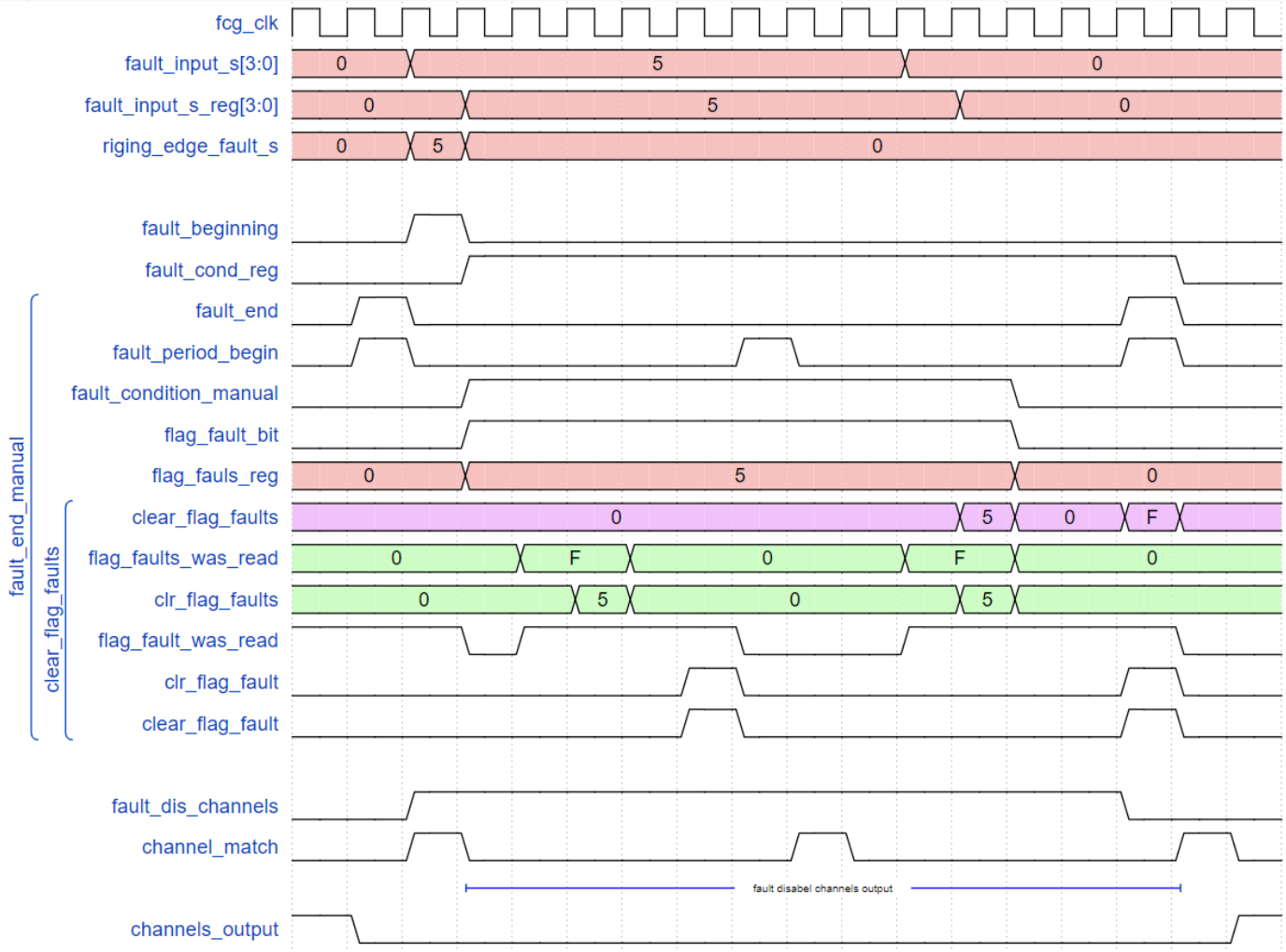
When the flag bit FAULTF and FAULTF(n) are all cleared and there is no fault input, the fault condition (fault_condition_manual) will be cleared.

The fault_cond_reg is used to record the fault state. It will be set when fault rising edge is detected and will be cleared at every new PWM cycle beginning if fault condition is cleared.

The fault disabling channel output will be enable at fault beginning and during fault state, except for fault ending time. Here the channel will be asserted when channel match (The FTU counter counts to the value of CV).

Flagchip

**Figure 105. Fault condition manual clearing**




### 49.5.9 Trigger Output Support

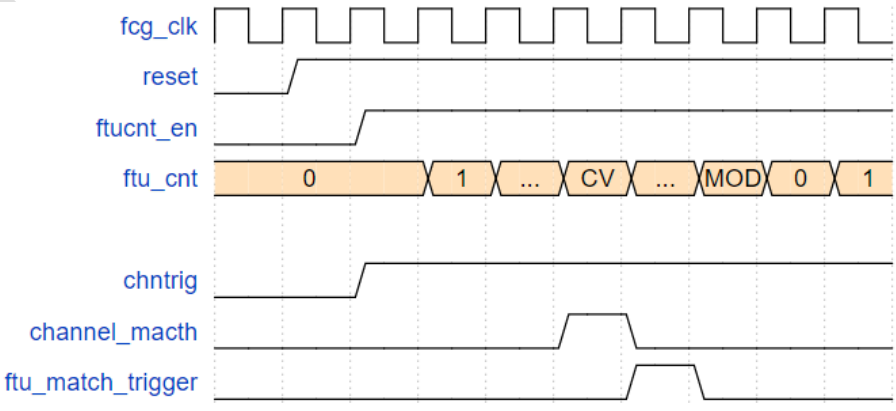
The FTU can generate multi triggers to trigger other peripherals on SoC. The width of these triggers is one FTU input clock.

#### *49.5.9.1 Match Trigger*

The FTU can generate a trigger when the channel (n) match occurs at the FTU Match Trigger output. To achieve this, the CHNTRIG must be set.

The match trigger can be cleared by UPDOWN_DIS in up-down counting mode.

**Figure 106. Channel match trigger**



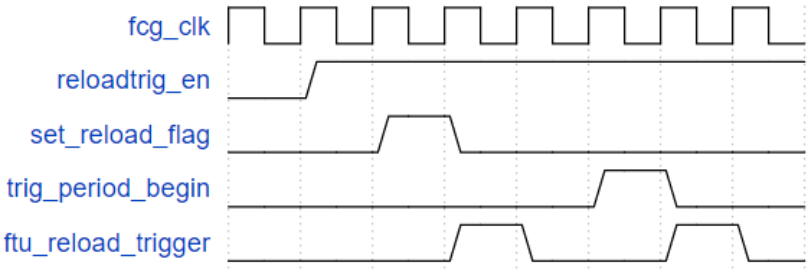
#### *49.5.9.2 Reload Trigger*

Reload Trigger is controlled by the bits RELOADTRIGEN and RTRIGR. The RELOADTRIGEN bit enables the Reload Trigger generation and the RTRIGR bit selects when the Reload Trigger is generated.

The reload trigger is generated in the following situation:

- FTU counter reaches a reload points (RELOADTRIGEN = 1 and RTRIGR = 1).
- FTU counter is updated with the CNTIN register value (RELOADTRIGEN = 1 and RTRIGR = 0).

**Figure 107. Reload trigger**




#### *49.5.9.3 Channel Trigger Output*

The channel trigger output provides a trigger signal, if the TRIGMODE is set, a trigger pulse is generated in the channel (n) output when a match occurs. It is only allowed to use trigger mode when channel (n) is in EPWM/CPWM and output compare mode.

### 49.5.10 Debug Mode

See the following table for FTU behavior during chip debug mode.

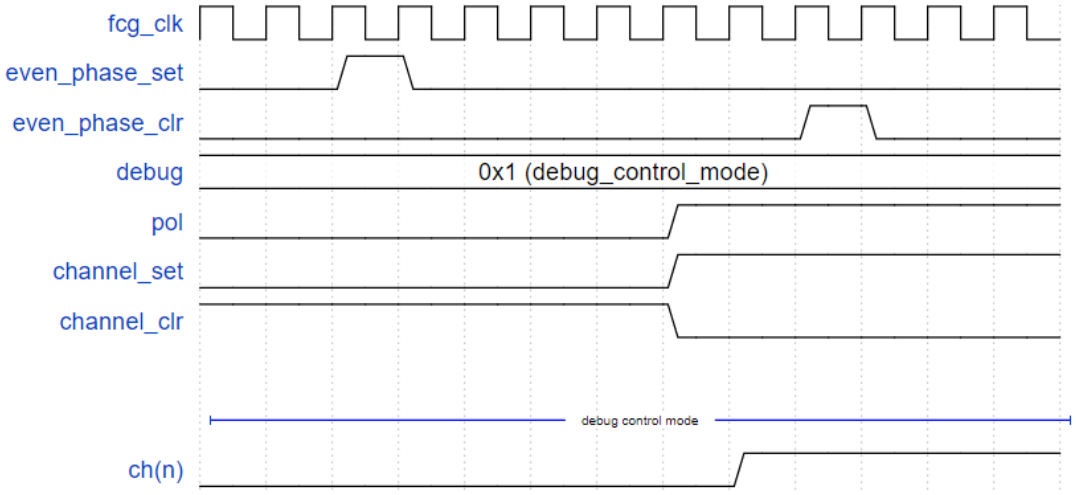
| DBGMODE | CHF bit | Writes to MOD, CNTIN, and C(n)V Registers |
|---|---|---|
| 00 | Can be set | Writes to these registers directly |
| 01 | Is not set | Writes to these registers directly |
| 10 | Is not set | Writes to these registers directly |
| 11 | Can be set | Functional mode |

The debug mode value is 2'b11 by default, which means FTU works in normal function mode. When debug mode value is 2'b00, the FTU counter will not start counting and its value will always be zero.

When debug mode value is 2'b01, the debug is in output control mode. When debug mode value is 2'b10, the debug is in output frozen mode.

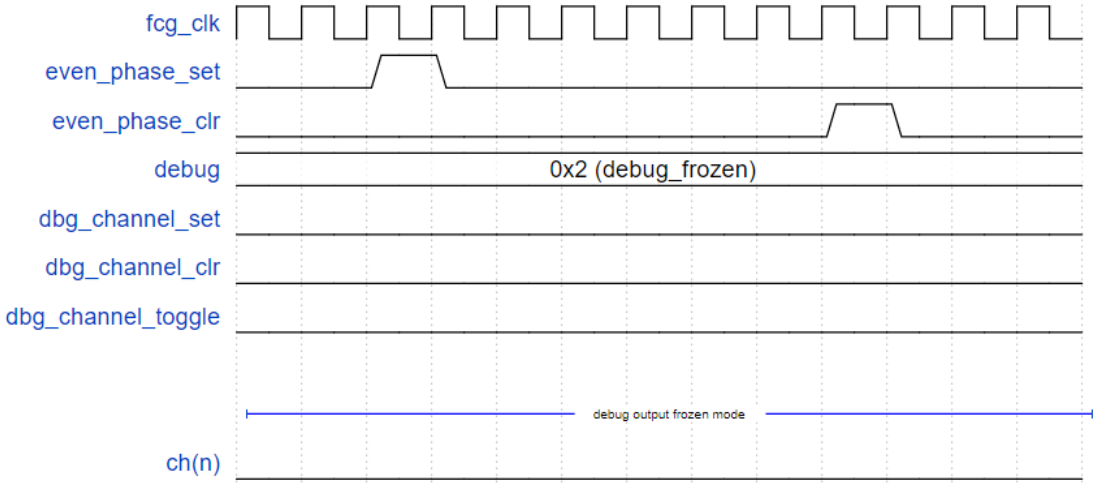
When the debug is in output control mode, the channel output is the value of POL:

**Figure 108. Debug Output Control**




When the debug is in output frozen mode, the channel output is zero:

**Figure 109. Debug Output Frozen**




### 49.5.11 Reload Points

The reload opportunities are:

1) the channel (n) match (CH(n)SEL = 1).
2) the FTU counter overflow
3) the FTU counter reload at up-down mode when CNTMIN = 1

If LDFQ = 0, then all reload opportunities are reload points.

Otherwise, the reload opportunity counter needs to get reach the value of LDFQ, and there are reload points, this counter is reset when there is a write to the register CNT. The RF bit is set at each reload point independent of the LDOK bit value.

#### *49.5.11.1 Enabling Global Time Base*

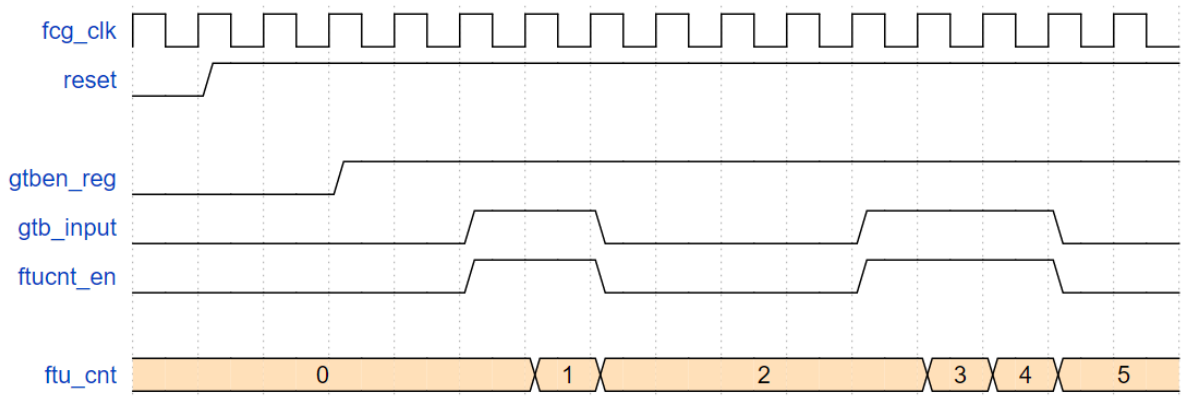
GTB is a function that allows the synchronization of multiple FTUs on a chip.

- If GTBEEN = 0, each FTU module works independently according to its configured mode.
- If GTBEEN = 1, the FTU counter update is enabled only when GTBIN is 1.

Write 1 to GTBEEN before enabling the clock of FTU to enable GTB.

The FTU counter will start count when GTBEEN (gtben_reg) is enabled and the GTB input (gtb_input) is high:

**Figure 110. GTB of FTU counter**




### 49.5.12 Channel Signal Related

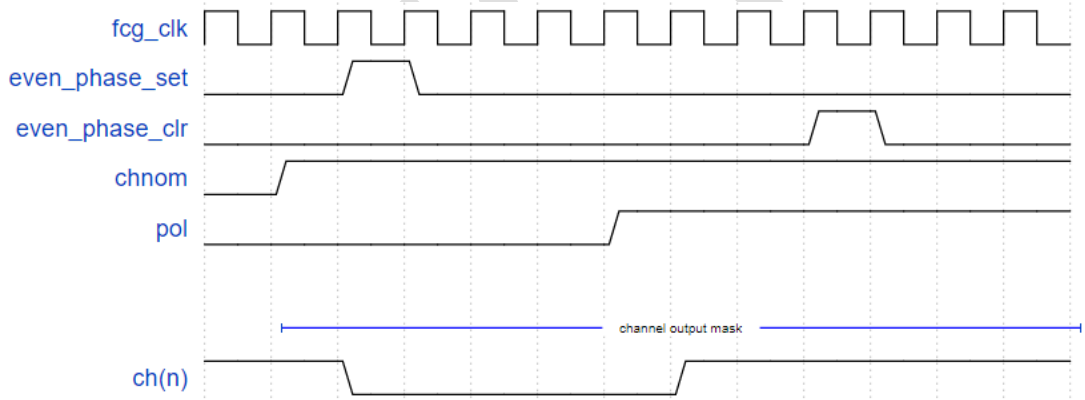
#### *49.5.12.1 Polarity Control*

In FTU, POL and FLTPOL control the polarity of channel and fault, when POL = 0, means the polarity is high, that is, logic 1 is a valid state and logic 0 is an invalid state, and when POL = 1, means it's an adverse polarity.

#### *49.5.12.2 Output Mask*

The output mask can be used to force channels output to their inactive state through software. If CH(n)OM = 1, then the channel (n) output is forced to its inactive state. If CH(n)OM = 0, then the channel (n) output is unaffected by the output mask.

**Figure 111. Channel Output Mask Control**



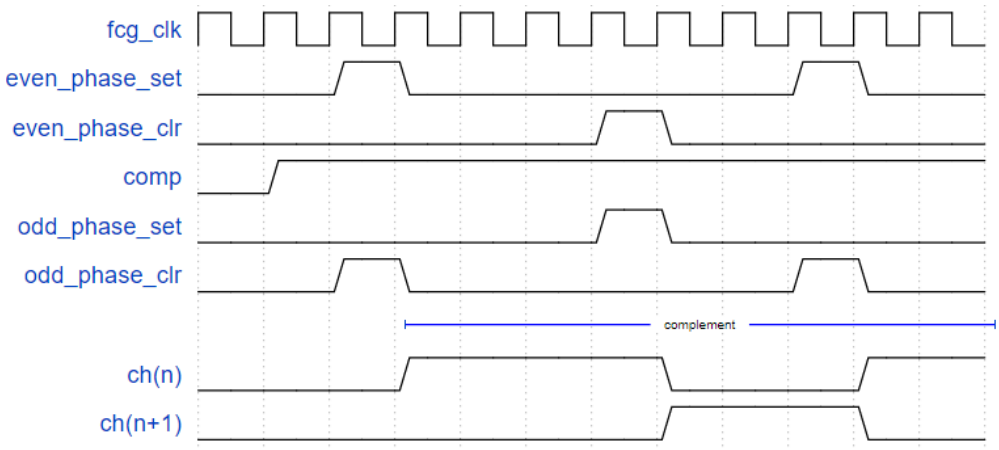

#### *49.5.12.3 Complement Output for Channel (n+1)*

If the COMP is asserted, the channel (n+1) output is complemented with channel n output, except for output compare toggle mode.

Ch0 means even channel (channel n) and Ch1 means odd channel (channel n+1) in pair of channels. If the COMP is asserted, the channel (n+1) output is complemented with channel n output, except for output compare toggle mode.

When comp is enabled, odd_phase_set changes to even_phase_clr and odd_phase_clr changes to even_phase_set, which make odd channel output is complemented with even channel output.

**Figure 112. Complement Control**

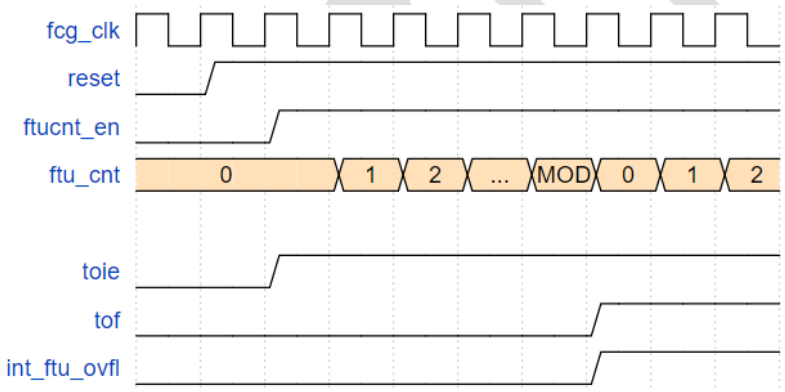



### 49.5.13 Interrupts

#### *49.5.13.1 Timer Overflow Interrupt*

The timer overflow interrupt is generated when (TOIE = 1) and (TOF = 1), TOF set when the counter changes from MOD to the next value.

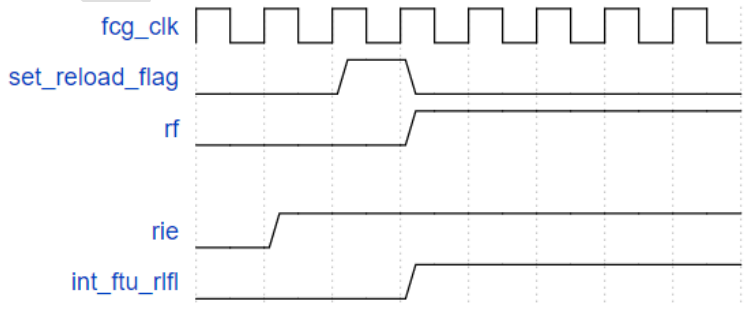
**Figure 113. FTU Counter interrupt generation**




#### *49.5.13.2 Reload Point Interrupt*

The Reload Point interrupt is generated when (RIE = 1) and (RF = 1).

**Figure 114. Reload point interrupt generation**

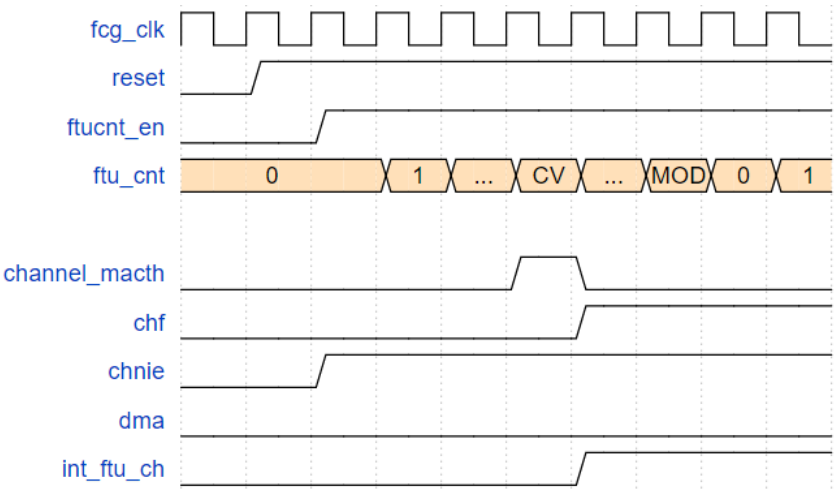



#### *49.5.13.3 Channel (n) Interrupt*

The channel (n) interrupt is generated when (CHIE = 1) and (CHF = 1).

The channel match interrupt can be cleared by UPDOWN_DIS in up-down counting mode.

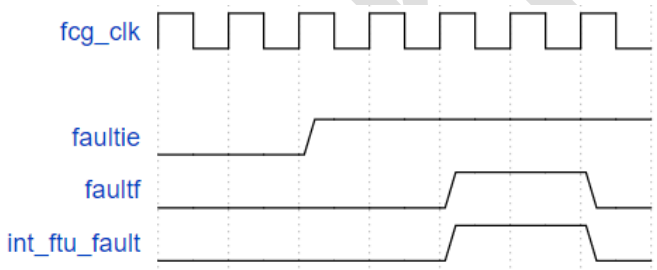
**Figure 115. Channel match interrupt generation**




#### *49.5.13.4 Fault Interrupt*

The fault interrupt is generated when (FAULTIE = 1) and (FAULTF = 1).

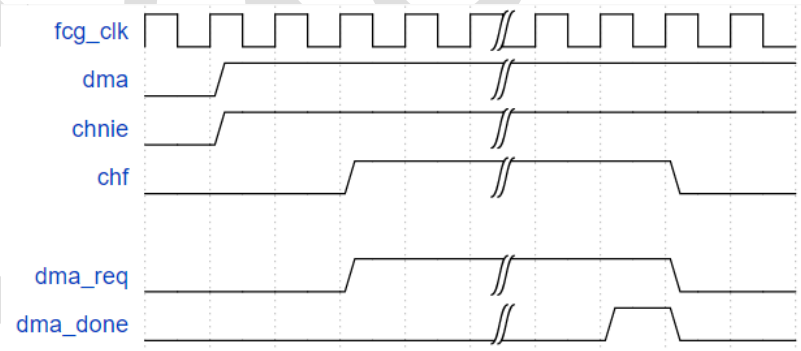
**Figure 116. Fault interrupt generation**




### 49.5.14 DMA

The channel DMA transfer request is generated if (CHF = 1) and (DMA = 1) and (CHIE = 1), and the CHF bit is cleared by channel DMA transfer done.

**Figure 117. DMA request**



## 49.6 Register Definition

FTU0 base address : 4005_C000h
FTU1 base address : 4005_D000h
FTU2 base address : 4005_E000h
FTU3 base address : 4005_F000h
FTU4 base address : 4045_F000h
FTU5 base address : 4046_0000h
FTU6 base address : 4046_1000h
FTU7 base address : 4046_2000h

**Register Overview - FTU**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | FTU_SC (Status and Control Register) | 32 | RW | 0000_0000h | page |
| **4h** | FTU_CNT (Counter Register) | 32 | RW | 0000_0000h | page |
| **8h** | FTU_MOD (Modulo Register) | 32 | RW | 0000_0000h | page |
| **Ch+n*8h** | FTU_CnSC (Channel n Status and Control Register) | 32 | RW | 0000_0000h | page |
| **10h+n*8h** | FTU_CnV (Channel n Value Register) | 32 | RW | 0000_0000h | page |
| **4Ch** | FTU_CNTIN (Counter Initial Value Register) | 32 | RW | 0000_0000h | page |
| **50h** | FTU_STATUS (Capture and Compare Status Register) | 32 | W0C | 0000_0000h | page |
| **54h** | FTU_MODE (Mode Selection Register) | 32 | RW | 0000_0004h | page |
| **58h** | FTU_SYNC (Synchronization Register) | 32 | RW | 0000_0000h | page |
| **5Ch** | FTU_OUTINIT (Initial State for Channels Output Register) | 32 | RW | 0000_0000h | page |
| **60h** | FTU_OUTMASK (Output Mask Register) | 32 | RW | 0000_0000h | page |
| **64h** | FTU_CHCTRL (Channel Control Register) | 32 | RW | 0000_0000h | page |
| **68h** | FTU_DEADTIME (Deadtime Configuration Register) | 32 | RW | 0000_0000h | page |
| **6Ch** | FTU_TRIGCONF (FTU Trigger Out Configuration Register) | 32 | RW | 0000_0000h | page |
| **70h** | FTU_POL (Polarity Register) | 32 | RW | 0000_0000h | page |
| **74h** | FTU_FMS (Fault Mode Status Register) | 32 | RW | 0000_0000h | page |
| **78h** | FTU_FILTER (Input Capture Filter Control Register) | 32 | RW | 0000_0000h | page |
| **7Ch** | FTU_FLTCTRL (Fault Control Register) | 32 | RW | 0000_0000h | page |
| **80h** | FTU_QDCTRL (Quadrature Decoder Control and Status Register) | 32 | RW | 0000_0000h | page |
| **84h** | FTU_CONF (Configuration Register) | 32 | RW | 0000_00C0h | page |
| **88h** | FTU_FLTPOL (FTU Fault Input Polarity Register) | 32 | RW | 0000_0000h | page |
| **8Ch** | FTU_SYNCONF (Synchronization Configuration Register) | 32 | RW | 0000_0000h | page |

| | | | | | |
|---|---|---|---|---|---|
| **90h** | FTU_INVCTRL (FTU Inverting Control Register) | 32 | RW | 0000_0000h | page |
| **94h** | FTU_SWOCTRL (FTU Software Output Control Register) | 32 | RW | 0000_0000h | page |
| **98h** | FTU_PWMLOAD (FTU PWM Load Register) | 32 | RW | 0000_0000h | page |
| **A0h+n*8h** | FTU_PAIRnDEADTIME (Pair (n) Deadtime Configuration Register) | 32 | RW | 0000_0000h | page |

### 49.6.1 FTU_SC (Status and Control Register)

SC contains global status and configuration for FTU.

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | UPDOWN_DIS | | 0 | | FLTPS | | | | CHNOUTEN | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TCKSEL | | 0 | | OBECTL | 0 | TOF | TOIE | RF | RIE | CPWMS | CLKS | | PS | | |
| W | | | | | | | w0c | | w0c | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_SC field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-30**<br>**UPDOWN_DIS** | RW | Disable channel match trigger/interrupt when count-up/down in CPWM/QUAD mode<br>This field is write protected (depends on WPDIS/WPEN).<br>00b - No effect<br>01b - Disable trigger/interrupt when count-down<br>10b - Disable trigger/interrupt when count-up<br>11b - Disable trigger/interrupt when count-up/down |
| **27-24**<br>**FLTPS** | RW | Filter Prescaler<br>Channel/fault/PHA/PHB input glitch filters clock prescaler.<br>Writing to the bits FLTPS has an immediate effect.<br><br>0000b - Divide by 1<br>0001b - Divide by 2<br>...<br>1111b - Divide by 16 |

| | | |
|---|---|---|
| **23-16**<br>**CHNOUTEN** | RW | Channel n Output Enable<br>This bit enables the channel output. It should be set to 0 if an input mode is used.<br>0b - Channel output is disabled.<br>1b - Channel output is enabled. |
| **15-14**<br>**TCKSEL** | RW | External Clock Select<br>Selects the external clock as the FTU function clock.<br>00b - FTU_TCLK0 pin as FTU external clock.<br>01b - FTU_TCLK1 pin as FTU external clock.<br>10b - FTU_TCLK2 pin as FTU external clock.<br>11b - No clock input |
| **11**<br>**OBECTL** | RW | Output Control Bit<br>Select the FTU output enable; should be ignored when FTU in input mode<br>0b - The FTU channel output is enabled by SC[CHnOUTEN].<br>In this configuration, the pad output state depends on FTU fault configuration.<br>1b - The FTU channel output is always enabled when FTU is in output mode. |
| **9**<br>**TOF** | W0C | Timer Overflow Flag<br>When the FTU counter passes the value in the MOD register, this bit will be asserted.<br>The TOF bit is cleared by reading the SC register while TOF is set and then writing a 0 to TOF bit. Writing a 1 to TOF has no effect.<br>If another FTU overflow occurs between the read and write operations, the write operation has no effect; therefore, TOF remains set indicating an overflow has occurred. In this case, a TOF interrupt request is not lost due to the clearing sequence for a previous TOF.<br>0b - FTU counter has not overflowed.<br>1b - FTU counter has overflowed. |
| **8**<br>**TOIE** | RW | Timer Overflow Interrupt Enable<br>0b - Disable TOF interrupts.<br>1b - Enable TOF interrupts. An interrupt is generated when TOF asserted. |
| **7**<br>**RF** | W0C | Reload Flag<br>The RF bit is set at each selected reload point.<br>The RF bit is cleared by reading the SC register while RF is set and then writing a 0 to RF bit. Writing 1 to RF has no effect.<br>If another selected reload point happens between the read and write operations, the write operation has no effect.<br>0b - A selected reload point did not happen.<br>1b - A selected reload point happened. |

| | | |
|---|---|---|
| **6**<br>**RIE** | RW | Reload Point Interrupt Enable<br>0b - Reload point interrupt is disabled.<br>1b - Reload point interrupt is enabled. |
| **5**<br>**CPWMS** | RW | Center-Aligned PWM Select<br>Selects CPWM mode. is write protected (depends on WPDIS/WPEN).<br>0b - FTU counter operates in Edge Align PWM mode.<br>1b - FTU counter operates in Center Align PWM mode. |
| **4-3**<br>**CLKS** | RW | Clock Source Selection<br>This field is write protected (depends on WPDIS/WPEN).<br>00b - No clock selected.<br>01b - Interface clock<br>10b - Reserved<br>11b - Functional clock |
| **2-0**<br>**PS** | RW | Prescale Setting<br>This field would affect the FTU counter clock frequency.<br>000b - Divide by 1<br>001b - Divide by 2<br>...<br>111b - Divide by 128 |

### 49.6.2 FTU_CNT (Counter Register)

The CNT register contains the FTU counter value.

Note: The counter width (16-bit or 24-bit) varies from instances, but is the same across registers CNT, MOD, CnV, and CNTIN.

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COUNT | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_CNT field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **15-0**<br>**COUNT** | RW | Counter Value |

### 49.6.3 FTU_MOD (Modulo Register)

The Modulo register contains the modulo value for the FTU counter. After the FTU counter reaches the modulo value, the overflow flag (TOF) set at the next clock cycle.

Writes to the MOD register are done on its write buffer, please see MOD register update for detail.

If FTUEN = 0, a write to SC register would cancel this buffer value, and need a new write operation to MOD.

It is recommended that initialize the FTU counter before writing to the MOD register to avoid confusion about when the first counter overflow.

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MOD | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_MOD field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**MOD** | RW | MOD<br>Modulo Value |

### 49.6.4 FTU_CnSC (Channel n Status and Control Register)

CSC contains channel control and status.

Offset : Ch+N*8h (N : 0 ~ 7)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ICEXP_NUM_ICM_ECNT | | | | | | | | 0 | | | | | | ICDPM_START | ICM_CONT |
| W | | | | | | | | | | | | | | ICM_SINGLE | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ICM_MODE | | | FDLYSEL | | CHOV | CHIS | TRIG MODE | CHF<br>w0c | CHIE | MSB | MSA | ELSB | ELSA | ICRST | DMA |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_CnSC field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**ICEXP_NUM_ICM_ECNT** | RW | Edge number counter in even channel (ICM_ECNT), read only (ro)<br>Expected number of edges in odd channel (ICEXP_NUM), read-write (rw)<br>This field is write protected (depends on WPDIS/WPEN). |
| **18**<br>**ICM_SINGLE** | WO | Re-start measurement when in single mode |
| **17**<br>**ICDPM_START** | RW | Start in ICDM & ICPM<br>(odd value is the same as even)<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - The channel starts measuring after the first edge is detected.<br>1b - The measurement starts immediately after activating the channel by ICDM = 1 or ICPM = 1. |
| **16**<br>**ICM_CONT** | RW | The measurement is single or continuous<br>(odd value is the same as even)<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - Single mode.<br>1b - Continuous mode. |
| **15-13**<br>**ICM_MODE** | RW | Channel Input Measurement Mode<br>(Odd value is the same as even)<br>This field is write protected (depends on WPDIS/WPEN).<br>001b - Select ICDM mode.<br>010b - Select ICPM mode.<br>011b - Select ICENM mode.<br>100b - Select ICEXPENM mode. |
| **12-11**<br>**FDLYSEL** | RW | Fault disable channel output delay value selection<br>This field is write protected (depends on WPDIS/WPEN).<br>00/11b - Select no delay.<br>01b - Select delay value 0.<br>10b - Select delay value 1. |

| | | |
|---|---|---|
| **10**<br>**CHOV** | RO | Channel (n) Output Value<br>The CHOV bit has the final value of the channel (n) output.<br>NOTE: The CHOV bit should be ignored when the channel (n) is not in an output mode.<br>0b - The channel (n) output is zero.<br>1b - The channel (n) output is one. |
| **9**<br>**CHIS** | RO | Channel (n) Input State<br>The CHIS bit has the value of the channel (n) input after the double-sampling or the filtering (if the channel (n) filter is enabled).<br>NOTE: The CHIS bit should be ignored when the channel (n) is not in an input mode.<br>0b - The channel (n) input is zero.<br>1b - The channel (n) input is one. |
| **8**<br>**TRIGMODE** | RW | Trigger Output Control<br>This bit controls the trigger generation.<br>This field is write protected (depends on WPDIS/WPEN)<br>0b - Channel outputs controlled by channel mode.<br>1b - If a match in the channel occurs, channel output will be a trigger. The trigger pulse width has one FTU interface cycle. |
| **7**<br>**CHF** | W0C | Channel (n) Flag<br>Set by hardware when an event occurs on the channel (n).<br>CHF is cleared by reading the CSC register while CHF is set and then writing a 0 to the CHF bit. Writing a 1 to CHF has no effect. |
| **6**<br>**CHIE** | RW | Channel (n) Interrupt Enable<br>0b - Disable channel (n) interrupt.<br>1b - Enable channel (n) interrupt. |
| **5**<br>**MSB** | RW | Channel (n) Mode Select<br>This field is write protected (depends on WPDIS/WPEN). |
| **4**<br>**MSA** | RW | Channel (n) Mode Select<br>This field is write protected (depends on WPDIS/WPEN). |
| **3**<br>**ELSB** | RW | Channel (n) Edge or Level Select<br>This field is write protected (depends on WPDIS/WPEN). |
| **2**<br>**ELSA** | RW | Channel (n) Edge or Level Select<br>This field is write protected (depends on WPDIS/WPEN). |

| | | |
|---|---|---|
| **1**<br>**ICRST** | RW | FTU counter reset is driven by the selected event of the channel (n) in the Input Capture mode.<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - FTU counter is not reset when the selected channel (n) input event is detected.<br>1b - FTU counter is reset when the selected channel (n) input event is detected. |
| **0**<br>**DMA** | RW | DMA Enable<br>0b - Disable DMA transfers.<br>1b - Enable DMA transfers. |

### 49.6.5 FTU_CnV (Channel n Value Register)

These registers contain the captured FTU counter value for the input modes or the match value for the output modes.

In Input Capture mode, any write to a CV register is ignored.

In output modes, writes to the CV register are done on its write buffer, please see CV register update for detail.

If FTUEN = 0, a write to SC register would cancel this buffer value, and need a new write operation to CV.

write to CV is ignored when there is not a valid channel mode.

Offset : 10h+N*8h (N : 0 ~ 7)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_CnV field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**VAL** | RW | Channel Value<br>Captured FTU counter value of the input modes or the match value for the output modes |

### 49.6.6 FTU_CNTIN (Counter Initial Value Register)

The Counter Initial Value register contains the initial value for the FTU counter.

Writing to the CNTIN register store the value into a buffer, please see CNTIN register update for detail.

When the FTU clock is initially selected, the FTU counter starts with the value 0x0000. To avoid this behavior, before the first

write to select the FTU clock, write the new value to the CNTIN register then initialize the FTU counter by writing any value to the CNT register.

Offset : 4Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INIT | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_CNTIN field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**INIT** | RW | INIT<br>Initial Value Of The FTU Counter |

### 49.6.7 FTU_STATUS (Capture and Compare Status Register)

STATUS register contains a copy of the status flag CHF bit in CSC for each FTU channel.

Each CHF bit in STATUS is a mirror of CHF bit in CSC.

All CHF bits can be cleared by reading STATUS followed by writing 0x00 to STATUS. Writing a 1 to CHF has no effect.

If another event occurs between the read and write operations, the write operation has no effect.

Offset : 50h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CHNF | | | | | | | |
| W | | | | | | | | | w0c | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **7-0**<br>**CHNF** | W0C | Channel n Flag<br>Bit(n) = 0b - No channel event has occurred on channel(n).<br>Bit(n) = 1b - A channel event has occurred on channel(n). |

### 49.6.8 FTU_MODE (Mode Selection Register)

This register contains the global enable bit for FTU-specific features, such as:

Fault control mode and interrupt

Synchronization

Write protection

Channel output initialization

These controls relate to all channels

Offset : 54h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | FAULTIE | FAULTM | | 0 | PWMSYNC | WPDIS | 0 | FTUEN |
| W | | | | | | | | | | | | | | | INIT | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**FTU_MODE field descriptions**

| Field | Type | Description |
|---|---|---|
| **7**<br>**FAULTIE** | RW | Fault Interrupt Enable<br>Enables the generation of an interrupt when a fault is detected by FTU and the FTU fault control is enabled.<br>0b - Fault control interrupt is disabled.<br>1b - Fault control interrupt is enabled. |

| | | |
|---|---|---|
| **6-5**<br>**FAULTM** | RW | Fault Control Mode<br>This field is write protected (depends on WPDIS/WPEN).<br>00b - Fault control is disabled for all channels.<br>01b - Fault control is enabled for even channels only (channels 0, 2, 4, and 6), and the selected mode is the manual fault clearing.<br>10b - Fault control is enabled for all channels, and the selected mode is the manual fault clearing.<br>11b - Fault control is enabled for all channels, and the selected mode is the automatic fault clearing. |
| **3**<br>**PWMSYNC** | RW | synchronization Mode<br>Selects which triggers can be used by MOD, CV, OUTMASK, and FTU counter synchronization.<br>0b - No restrictions. Software and hardware triggers can be used by MOD, CV, OUTMASK, and FTU counter synchronization.<br>1b - Software trigger can only be used by MOD and CV synchronization, and hardware triggers can only be used by OUTMASK and FTU counter synchronization. |
| **2**<br>**WPDIS** | RW | Write Protection Disable<br>The WPDIS bit is the negation of the WPEN bit. WPDIS is cleared when 1 is written to WPEN. WPDIS is set when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPDIS has no effect.<br>0b - Write protection is enabled, write protected bits cannot be written.<br>1b - Write protection is disabled, write protected bits can be written. |
| **1**<br>**INIT** | WO | Initialize The Channels Output<br>When a 1 is written to INIT bit, the channels output is initialized to the state of their corresponding bit defined in the OUTINIT register.<br>Writing a 0 to INIT bit has no effect.<br>The INIT bit is always read as 0. |
| **0**<br>**FTUEN** | RW | FTU Enable<br>This field is write protected (depends on WPDIS/WPEN).<br>This field defines different free running counters and synchronization behaviors. For more information, refer to Function Description chapter. |

### 49.6.9 FTU_SYNC (Synchronization Register)

This register configures the synchronization.

A synchronization event can perform the synchronized update of MOD, CV, and OUTMASK registers with the value of their write buffer and the FTU counter initialization.

NOTE:

The software trigger, SWSYNC bit, and hardware triggers TRIG0, TRIG1, and TRIG2 bits have a potential conflict. Use only hardware or software triggers but not both at the same time, otherwise unpredictable behavior is likely to happen.

The selection of the loading point, CNTMAX and CNTMIN bits, is intended to provide the update of MOD, CNTIN, and CV registers across all enabled channels simultaneously.

Offset : 58h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | SWSYNC | TRIG2 | TRIG1 | TRIG0 | SYNCHOM | REINIT | CNTMAX | CNTMIN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_SYNC field descriptions**

| Field | Type | Description |
|---|---|---|
| **7**<br>**SWSYNC** | RW | Synchronization Software Trigger<br>Selects the software trigger as the synchronization trigger. The software trigger happens when a 1 is written to SWSYNC bit.<br>0b - Software trigger is not selected.<br>1b - Software trigger is selected. |
| **6**<br>**TRIG2** | RW | Synchronization Hardware Trigger 2<br>Enables hardware trigger 2 to the synchronization. Hardware trigger 2 happens when a rising edge is detected at the trigger 2 input signal.<br>0b - Trigger is disabled.<br>1b - Trigger is enabled. |
| **5**<br>**TRIG1** | RW | Synchronization Hardware Trigger 1<br>Enables hardware trigger 1 to the synchronization. Hardware trigger 1 happens when a rising edge is detected at the trigger 1 input signal.<br>0b - Trigger is disabled.<br>1b - Trigger is enabled. |
| **4**<br>**TRIG0** | RW | Synchronization Hardware Trigger 0<br>Enables hardware trigger 0 to the synchronization. Hardware trigger 0 occurs when a rising edge is detected at the trigger 0 input signal.<br>0b - Trigger is disabled.<br>1b - Trigger is enabled. |

| | | |
|---|---|---|
| **3**<br>**SYNCHOM** | RW | Output Mask Synchronization<br>Selects when the OUTMASK register is updated with the value of its buffer.<br>0b - OUTMASK register is updated with the value of its buffer in all rising edges of the FTU input clock.<br>1b - OUTMASK register is updated with the value of its buffer only by the synchronization. |
| **2**<br>**REINIT** | RW | FTU Counter Reinitialization by Synchronization<br>Determines if the FTU counter is reinitialized when the selected trigger for the synchronization is detected.<br>0b - FTU counter continues to count normally.<br>1b - FTU counter is updated with its initial value when the selected trigger is detected. |
| **1**<br>**CNTMAX** | RW | Maximum Loading Point Enable<br>Selects the maximum loading point to synchronization. If CNTMAX is 1, the selected loading point is when the FTU counter reaches MOD.<br>0b - The maximum loading point is disabled.<br>1b - The maximum loading point is enabled. |
| **0**<br>**CNTMIN** | RW | Minimum Loading Point Enable<br>Selects the minimum loading point to synchronization. If CNTMIN is 1, the selected loading point is when the FTU counter reaches CNTIN.<br>0b - The minimum loading point is disabled.<br>1b - The minimum loading point is enabled. |

### 49.6.10 FTU_OUTINIT (Initial State for Channels Output Register)

Offset : 5Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CHNOI | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_OUTINIT field descriptions**

| Field | Type | Description |
|---|---|---|
| **7-0**<br>**CHNOI** | RW | Channel n Output Initialization Value<br>Selects the value that is forced into the channel output when the initialization occurs.<br>Bit(n) = 0b - The initialization value of channel (n) is 0.<br>Bit(n) = 1b - The initialization value of channel (n) is 1. |

### 49.6.11 FTU_OUTMASK (Output Mask Register)

This register provides a mask for each FTU channel.

Any write to the OUTMASK register, stores the value in its write buffer, please see OUTMASK register update for detail.

Output Mask bits must not be set for trigger mode.

Offset : 60h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CHNOM | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_OUTMASK field descriptions**

| Field | Type | Description |
|---|---|---|
| **7-0**<br>**CHNOM** | RW | Channel n Output Mask<br>Defines if the channel (n) output is masked or unmasked.<br>Bit(n) = 0b - Channel (n) output is not masked. It continues to operate normally.<br>Bit(n) = 1b - Channel (n) output is masked. It is forced to its inactive state. |

### 49.6.12 FTU_CHCTRL (Channel Control Register)

The function of bit 23-16, bit 15-8, and bit 7-0 is the same as that of bit 31-24.

Offset : 64h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EPHASE3 | FAULTEN3 | SYNCEN3 | DTEN3 | 0 | | COMP3 | PHASE3 | EPHASE2 | FAULTEN2 | SYNCEN2 | DTEN2 | 0 | | COMP2 | PHASE2 |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EPHASE1 | FAULTEN1 | SYNCEN1 | DTEN1 | 0 | | COMP1 | PHASE1 | EPHASE0 | FAULTEN0 | SYNCEN0 | DTEN0 | 0 | | COMP0 | PHASE0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_CHCTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**EPHASE3** | RW | Enhanced Phase Shift Mode Enable for n = 6<br>Enable Enhanced Phase Shift mode<br>0b - The Enhanced Phase Shift is disabled<br>1b - The Enhanced Phase Shift is enabled |
| **30**<br>**FAULTEN3** | RW | Fault Control Enable for n = 6<br>Enables the fault control in channels (n) and (n+1).<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - The fault control in this pair of channels is disabled.<br>1b - The fault control in this pair of channels is enabled. |
| **29**<br>**SYNCEN3** | RW | Synchronization Enable for n = 6<br>Enables synchronization of registers C(n)V and C(n+1)V.<br>0b - The synchronization in this pair of channels is disabled.<br>1b - The synchronization in this pair of channels is enabled. |
| **28**<br>**DTEN3** | RW | Deadtime Enable for n = 6<br>Enables the deadtime insertion in the channels (n) and (n+1).<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - The deadtime insertion in this pair of channels is disabled.<br>1b - The deadtime insertion in this pair of channels is enabled. |
| **25**<br>**COMP3** | RW | Complement of Channel (n) for n = 6<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - do not complement channel (n+1) output with channel (n) output.<br>1b - The channel (n+1) output is the complement of the channel (n) output. |
| **24**<br>**PHASE3** | RW | Phase Shift Mode Enable for n = 6<br>Enable Phase Shift mode<br>0b - The Phase Shift is disabled<br>1b - The Phase Shift is enabled |

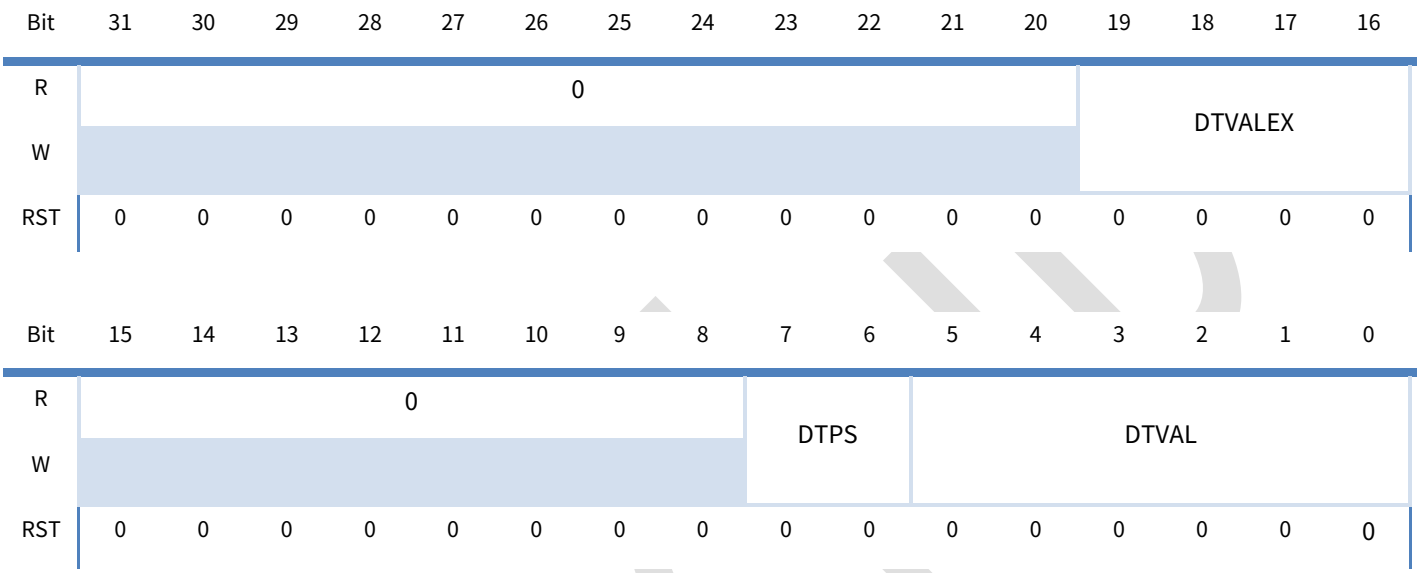
| | | |
|---|---|---|
| **23**<br>**EPHASE2** | RW | Enhanced Phase Shift Mode Enable for n = 4<br>Enable Enhanced Phase Shift mode<br>0b - The Enhanced Phase Shift is disabled<br>1b - The Enhanced Phase Shift is enabled |
| **22**<br>**FAULTEN2** | RW | Fault Control Enable for n = 4<br>Enables the fault control in channels (n) and (n+1).<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - The fault control in this pair of channels is disabled.<br>1b - The fault control in this pair of channels is enabled. |
| **21**<br>**SYNCEN2** | RW | Synchronization Enable for n = 4<br>Enables synchronization of registers C(n)V and C(n+1)V.<br>0b - The synchronization in this pair of channels is disabled.<br>1b - The synchronization in this pair of channels is enabled. |
| **20**<br>**DTEN2** | RW | Deadtime Enable for n = 4<br>Enables the deadtime insertion in the channels (n) and (n+1).<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - The deadtime insertion in this pair of channels is disabled.<br>1b - The deadtime insertion in this pair of channels is enabled. |
| **17**<br>**COMP2** | RW | Complement of Channel (n) for n = 4<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - Do not complement channel (n+1) output with channel (n) output.<br>1b - The channel (n+1) output is the complement of the channel (n) output. |
| **16**<br>**PHASE2** | RW | Phase Shift Mode Enable for n = 4<br>Enable Phase Shift mode<br>0b - The Phase Shift is disabled<br>1b - The Phase Shift is enabled |
| **15**<br>**EPHASE1** | RW | Enhanced Phase Shift Mode Enable for n = 2<br>Enable Enhanced Phase Shift mode<br>0b - The Enhanced Phase Shift is disabled<br>1b - The Enhanced Phase Shift is enabled |
| **14**<br>**FAULTEN1** | RW | Fault Control Enable for n = 2<br>Enables the fault control in channels (n) and (n+1).<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - The fault control in this pair of channels is disabled.<br>1b - The fault control in this pair of channels is enabled. |
| **13**<br>**SYNCEN1** | RW | Synchronization Enable for n = 2<br>Enables synchronization of registers C(n)V and C(n+1)V.<br>0b - The synchronization in this pair of channels is disabled.<br>1b - The synchronization in this pair of channels is enabled. |

| | | |
|---|---|---|
| **12**<br>**DTEN1** | RW | Deadtime Enable for n = 2<br>Enables the deadtime insertion in the channels (n) and (n+1).<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - The deadtime insertion in this pair of channels is disabled.<br>1b - The deadtime insertion in this pair of channels is enabled. |
| **9**<br>**COMP1** | RW | Complement of Channel (n) for n = 2<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - Do not complement channel (n+1) output with channel (n) output.<br>1b - The channel (n+1) output is the complement of the channel (n) output. |
| **8**<br>**PHASE1** | RW | Phase Shift Mode Enable for n = 2<br>Enable Phase Shift mode<br>0b - The Phase Shift is disabled<br>1b - The Phase Shift is enabled |
| **7**<br>**EPHASE0** | RW | Enhanced Phase Shift Mode Enable for n = 0<br>Enable Enhanced Phase Shift mode<br>0b - The Enhanced Phase Shift is disabled<br>1b - The Enhanced Phase Shift is enabled |
| **6**<br>**FAULTEN0** | RW | Fault Control Enable for n = 0<br>Enables the fault control in channels (n) and (n+1).<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - The fault control in this pair of channels is disabled.<br>1b - The fault control in this pair of channels is enabled. |
| **5**<br>**SYNCEN0** | RW | Synchronization Enable for n = 0<br>Enables synchronization of registers C(n)V and C(n+1)V.<br>0b - The synchronization in this pair of channels is disabled.<br>1b - The synchronization in this pair of channels is enabled. |
| **4**<br>**DTEN0** | RW | Deadtime Enable for n = 0<br>Enables the deadtime insertion in the channels (n) and (n+1).<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - The deadtime insertion in this pair of channels is disabled.<br>1b - The deadtime insertion in this pair of channels is enabled. |
| **1**<br>**COMP0** | RW | Complement of Channel (n) for n = 0<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - do not complement channel (n+1) output with channel (n) output.<br>1b - The channel (n+1) output is the complement of the channel (n) output. |

| | | |
|---|---|---|
| **0**<br>**PHASE0** | RW | Phase Shift Mode Enable for n = 0<br>Enable Phase Shift mode<br>0b - The Phase Shift is disabled<br>1b - The Phase Shift is enabled |

### 49.6.13 FTU_DEADTIME (Deadtime Configuration Register)

Offset : 68h




**FTU_DEADTIME field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16**<br>**DTVALEX** | RW | Extended Deadtime Value<br>This field is a bit extension of the DTVAL field. It defines the 4 most significant bits of the deadtime value. The maximum deadtime value is extended to 1023 using the concatenation {DTVALEX, DTVAL}.<br>Deadtime insert value = (DTPS × {DTVALEX, DTVAL}).<br>This field is write protected (depends on WPDIS/WPEN).<br>NOTE: If full compatibility is needed with previous software versions, write 0 to DTVALEX bits. |
| **7-6**<br>**DTPS** | RW | Deadtime Prescaler Value<br>Selects the division factor of the FTU input clock. This prescaled clock is used by the deadtime counter. This field is write protected (depends on WPDIS/WPEN).<br>0xb - Divide the FTU input clock by 1.<br>10b - Divide the FTU input clock by 4.<br>11b - Divide the FTU input clock by 16. |
| **5-0**<br>**DTVAL** | RW | Deadtime Value<br>Selects the deadtime value.<br>This field is write protected (depends on WPDIS/WPEN). |

### 49.6.14 FTU_TRIGCONF (FTU Trigger Out Configuration Register)

Offset : 6Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | CH7TRIG | CH6TRIG | TRIGF | RELOADTRIGEN | CH1TRIG | CH0TRIG | CH5TRIG | CH4TRIG | CH3TRIG | CH2TRIG |
| W | | | | | | | | | w0c | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_TRIGCONF field descriptions**

| Field | Type | Description |
|---|---|---|
| **9**<br>**CH7TRIG** | RW | Channel 7 Match Trigger Enable<br>Enables the generation of the match Trigger when FTU counter = C7V.<br>0b - The generation of this match Trigger is disabled.<br>1b - The generation of this match Trigger is enabled. |
| **8**<br>**CH6TRIG** | RW | Channel 6 Match Trigger Enable<br>Enables the generation of the match Trigger when FTU counter = C6V.<br>0b - The generation of this match Trigger is disabled.<br>1b - The generation of this match Trigger is enabled. |
| **7**<br>**TRIGF** | W0C | Channel Trigger Flag<br>Set by hardware when a channel trigger is generated. Clear TRIGF by reading TRIGCONF while TRIGF is set and then writing a 0 to TRIGF. Writing a 1 to TRIGF has no effect.<br>If another channel trigger is generated before the clearing sequence is completed, the sequence is reset so TRIGF remains set after the clear sequence is completed for the earlier TRIGF.<br>0b - No channel trigger was generated.<br>1b - A channel trigger was generated. |
| **6**<br>**RELOADTRIGEN** | RW | Reload Trigger Enable<br>Enables the generation of the trigger when the FTU counter is equal to the CNTIN register.<br>0b - The generation of reload trigger is disabled.<br>1b - The generation of reload trigger is enabled. |

| | | |
|---|---|---|
| **5**<br>**CH1TRIG** | RW | Channel 1 Match Trigger Enable<br>Enables the generation of the match Trigger when FTU counter = C1V.<br>0b - The generation of this match Trigger is disabled.<br>1b - The generation of this match Trigger is enabled. |
| **4**<br>**CH0TRIG** | RW | Channel 0 Match Trigger Enable<br>Enables the generation of the match Trigger when FTU counter = C0V.<br>0b - The generation of this match Trigger is disabled.<br>1b - The generation of this match Trigger is enabled. |
| **3**<br>**CH5TRIG** | RW | Channel 5 Match Trigger Enable<br>Enables the generation of the match Trigger when FTU counter = C5V.<br>0b - The generation of this match Trigger is disabled.<br>1b - The generation of this match Trigger is enabled. |
| **2**<br>**CH4TRIG** | RW | Channel 4 Match Trigger Enable<br>Enables the generation of the match Trigger when FTU counter = C4V.<br>0b - The generation of this match Trigger is disabled.<br>1b - The generation of this match Trigger is enabled. |
| **1**<br>**CH3TRIG** | RW | Channel 3 Match Trigger Enable<br>Enables the generation of the match Trigger when FTU counter = C3V.<br>0b - The generation of this match Trigger is disabled.<br>1b - The generation of this match Trigger is enabled. |
| **0**<br>**CH2TRIG** | RW | Channel 2 Match Trigger Enable<br>Enables the generation of the match Trigger when FTU counter = C2V.<br>0b - The generation of this match Trigger is disabled.<br>1b - The generation of this match Trigger is enabled. |

### 49.6.15 FTU_POL (Polarity Register)

Offset : 70h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | POLN | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_POL field descriptions**

| Field | Type | Description |
|---|---|---|
| **7-0**<br>**POLN** | RW | Channel n Polarity<br>Defines the polarity of the channel output.<br>This field is write protected (depends on WPDIS/WPEN).<br>Bit(n) = 0b - The channel (n) polarity is active high.<br>Bit(n) = 1b - The channel (n) polarity is active low. |

### 49.6.16 FTU_FMS (Fault Mode Status Register)

Offset : 74h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | FAULTF | WPEN | FAULTIN | 0 | | | FAULTF1 | FAULTF0 |
| W | | | | | | | | | w0c | | | | | | w0c | w0c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_FMS field descriptions**

| Field | Type | Description |
|---|---|---|
| **7**<br>**FAULTF** | W0C | Fault Detection Flag<br>Represents the logic OR of the FAULTF bit of each enabled fault input. Clear FAULTF by reading the FMS register while FAULTF is set and then writing a 0 to FAULTF while there is no existing fault condition at the enabled fault inputs. Writing a 1 to FAULTF has no effect. |
| **6**<br>**WPEN** | RW | Write Protection Enable<br>The WPEN bit is the negation of the WPDIS bit. WPEN is set when 1 is written to it. WPEN is cleared when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPEN has no effect.<br>0b - Write protection is disabled. Write protected bits can be written.<br>1b - Write protection is enabled. Write protected bits cannot be written. |

| | | |
|---|---|---|
| **5**<br>**FAULTIN** | RO | Fault Inputs<br>Represents the logic OR of the enabled fault inputs after their filter (if their filter is enabled) when fault control is enabled.<br>0b - The logic OR of the enabled fault inputs is 0.<br>1b - The logic OR of the enabled fault inputs is 1. |
| **1**<br>**FAULTF1** | W0C | Fault Detection Flag 1<br>0b - No fault condition was detected at the fault input.<br>1b - A fault condition was detected at the fault input. |
| **0**<br>**FAULTF0** | W0C | Fault Detection Flag 0<br>0b - No fault condition was detected at the fault input.<br>1b - A fault condition was detected at the fault input. |

### 49.6.17 FTU_FILTER (Input Capture Filter Control Register)

Offset : 78h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CH3FVAL | | | | CH2FVAL | | | | CH1FVAL | | | | CH0FVAL | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_FILTER field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-12**<br>**CH3FVAL** | RW | Channel 3 Input Filter<br>Selects the filter value for the channel input. The filter is disabled when the value is zero.<br>The filter value is CH3FVAL*4. |
| **11-8**<br>**CH2FVAL** | RW | Channel 2 Input Filter<br>Selects the filter value for the channel input. The filter is disabled when the value is zero.<br>The filter value is CH2FVAL*4. |

| | | |
|---|---|---|
| **7-4**<br>**CH1FVAL** | RW | Channel 1 Input Filter<br>Selects the filter value for the channel input. The filter is disabled when the value is zero.<br>The filter value is CH1FVAL*4. |
| **3-0**<br>**CH0FVAL** | RW | Channel 0 Input Filter<br>Selects the filter value for the channel input. The filter is disabled when the value is zero.<br>The filter value is CH0FVAL*4. |

### 49.6.18 FTU_FLTCTRL (Fault Control Register)

Offset : 7Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | FDLYV0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FSTATE | 0 | | | FFVAL | | | | 0 | | FLT1GFEN | FLT0GFEN | 0 | | FAULT1EN | FAULT0EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_FLTCTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-16**<br>**FDLYV0** | RW | Fault Disable Channel Output Delay Value 0<br>This field is write protected (depends on WPDIS/WPEN). |
| **15**<br>**FSTATE** | RW | Fault Output State<br>This configuration allows to put the FTU outputs to pre-define state when a fault event is ongoing.<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - FTU outputs will be placed into safe values when fault events in ongoing (defined by POL bits).<br>1b - FTU outputs will be tri-stated when fault event is ongoing |

| | | |
|---|---|---|
| **11-8**<br>**FFVAL** | RW | Fault Input Filter<br>Selects the filter value for the fault inputs.<br>The fault filter is disabled when the value is zero.<br>NOTE: Writing to this field has immediate effect and must be done only when the fault control or all fault inputs are disabled. Failure to do this could result in a missing fault detection. |
| **5**<br>**FLT1GFEN** | RW | Fault Input 1 Glitch Filter Enable<br>0b - Fault input filter is disabled.<br>1b - Fault input filter is enabled. |
| **4**<br>**FLT0GFEN** | RW | Fault Input 0 Glitch Filter Enable<br>0b - Fault input filter is disabled.<br>1b - Fault input filter is enabled. |
| **1**<br>**FAULT1EN** | RW | Fault Input 1 Enable Enables the fault input.<br>0b - Fault input is disabled.<br>1b - Fault input is enabled. |
| **0**<br>**FAULT0EN** | RW | Fault Input 0 Enable Enables the fault input.<br>0b - Fault input is disabled.<br>1b - Fault input is enabled. |

### 49.6.19 FTU_QDCTRL (Quadrature Decoder Control and Status Register)

The register is only valid when Quadrature Decoder is supported. In the FC7240 family, only valid for FTU1, FTU2, FTU4, and FTU5.

Offset : 80h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | PHAGFEN | PHBGFEN | PHAPOL | PHBPOL | QUADMODE | QUADIR | TOFDIR | QUADEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_QDCTRL field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **7**<br>**PHAGFEN** | RW | Phase A Glitch Filter Enable<br>Enables the glitch filter for the quadrature decoder phase A input. The filter value for the phase A input is defined by the CH0FVAL field of FILTER. The phase A glitch filter is disabled when CH0FVAL is zero.<br>0b - Phase A input filter is disabled.<br>1b - Phase A input filter is enabled. |
| **6**<br>**PHBGFEN** | RW | Phase B Glitch Filter Enable<br>Enables the filter for the quadrature decoder phase B input. The filter value for the phase B input is defined by the CH1FVAL field of FILTER. The phase B filter is also disabled when CH1FVAL is zero. |
| **5**<br>**PHAPOL** | RW | Phase A Input Polarity<br>Selects the polarity for the quadrature decoder phase A input.<br>0b - Normal polarity.<br>1b - Inverted polarity. Phase A input signal is inverted before identifying the rising and falling edges of this signal. |
| **4**<br>**PHBPOL** | RW | Phase B Input Polarity<br>Selects the polarity for the quadrature decoder phase B input. |
| **3**<br>**QUADMODE** | RW | Quadrature Decoder Mode<br>Selects the encoding mode used in the Quadrature Decoder mode.<br>0b - Phase A and phase B encoding mode.<br>1b - Count and direction encoding mode. |
| **2**<br>**QUADIR** | RO | FTU Counter Direction In Quadrature Decoder Mode<br>Indicates the counting direction.<br>0b - FTU counter decrement.<br>1b - FTU counter increment. |
| **1**<br>**TOFDIR** | RO | Timer Overflow Direction In Quadrature Decoder Mode<br>Indicates if the TOF bit was set on the top or the bottom of counting.<br>0b - TOF bit was set on the bottom of counting. FTU counter decrement and counter changes from CNTIN to MOD.<br>1b - TOF bit was set on the top of counting. FTU counter increment and counter changes from MOD to CNTIN. |
| **0**<br>**QUADEN** | RW | Quadrature Decoder Mode Enable<br>Enables the Quadrature Decoder mode. In this mode, the phase A and B input signals control the FTU counter. this mode has precedence over the other modes.<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - Quadrature Decoder mode is disabled.<br>1b - Quadrature Decoder mode is enabled. |

### 49.6.20 FTU_CONF (Configuration Register)

Offset : 84h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | RTRIGR | 0 | GTBEEN | 0 | DBG | | 0 | LDFQ | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_CONF field descriptions**

| Field | Type | Description |
|---|---|---|
| **11**<br>**RTRIGR** | RW | Reload trigger on Reload Point<br>0b - Reload trigger is generated on counter overflow events.<br>1b - Reload trigger is generated when a reload point is reached. |
| **9**<br>**GTBEEN** | RW | Global Time Base Enable<br>Configures the FTU to use an external global time base signal.<br>0b - Use of an external global time base is disabled.<br>1b - Use of an external global time base is enabled. |
| **7-6**<br>**DBG** | RW | Debug Mode<br>Selects the FTU behavior in Debug mode.<br>00b - FTU counter stopped, channel works as function mode.<br>01b - FTU counter stopped, channel output force to safe state.<br>10b - FTU counter stopped, channel output is frozen<br>11b - FTU counter works as function mode, channel works as function mode |
| **4-0**<br>**LDFQ** | RW | Frequency of the Reload Opportunities<br>The LDFQ bits define the number of enabled reload opportunities that should happen until an enabled reload opportunity becomes a reload point. See Reload Points<br>0: All reload opportunities are reload points<br>1: There is a reload point for each 2 reload opportunities.<br>2: There is a reload point for each 3 reload opportunities.<br>...<br>31: There is a reload point for each 32 reload opportunities |

### 49.6.21 FTU_FLTPOL (FTU Fault Input Polarity Register)

Offset : 88h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | FDLYV1 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | FLT3POL | FLT2POL | FLT1POL | FLT0POL |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_FLTPOL field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-16**<br>**FDLYV1** | RW | Fault Disable Channel Output Delay Value 1<br>This field is write protected (depends on WPDIS/WPEN). |
| **3**<br>**FLT3POL** | RW | Fault Input 3 Polarity<br>Defines the polarity of the fault input.<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - The fault input polarity is active high. 1 at the fault input indicates a fault.<br>1b - The fault input polarity is active low. 0 at the fault input indicates a fault. |
| **2**<br>**FLT2POL** | RW | Fault Input 2 Polarity<br>Defines the polarity of the fault input.<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - The fault input polarity is active high. 1 at the fault input indicates a fault.<br>1b - The fault input polarity is active low. 0 at the fault input indicates a fault. |
| **1**<br>**FLT1POL** | RW | Fault Input 1 Polarity<br>Defines the polarity of the fault input.<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - The fault input polarity is active high. 1 at the fault input indicates a fault.<br>1b - The fault input polarity is active low. 0 at the fault input indicates a fault. |

| | | |
|---|---|---|
| **0**<br>**FLT0POL** | RW | Fault Input 0 Polarity<br>Defines the polarity of the fault input.<br>This field is write protected (depends on WPDIS/WPEN).<br>0b - The fault input polarity is active high. 1 at the fault input indicates a fault.<br>1b - The fault input polarity is active low. 0 at the fault input indicates a fault. |

### 49.6.22 FTU_SYNCONF (Synchronization Configuration Register)

Offset : 8Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | CNTINC | 0 | HWTRIGMODE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_SYNCONF field descriptions**

| Field | Type | Description |
|---|---|---|
| **2**<br>**CNTINC** | RW | CNTIN Register Synchronization<br>0b - CNTIN register is updated with its buffer value at all rising edges of FTU input clock.<br>1b - CNTIN register is updated with its buffer value by the synchronization. |
| **0**<br>**HWTRIGMODE** | RW | Hardware Trigger Mode<br>0b - FTU clears the Sync[TRIG 0/1/2] bit when the hardware trigger j is detected, where n = 0, 1, 2.<br>1b - FTU does not clear the Sync[TRIG 0/1/2] bit when the hardware trigger j is detected, where n = 0, 1, 2. |

### 49.6.23 FTU_INVCTRL (FTU Inverting Control Register)

Offset : 90h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | INV3EN | INV2EN | INV1EN | INV0EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_INVCTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **3**<br>**INV3EN** | RW | Pair Channels 3 Inverting Enable<br>0b - Inverting is disabled.<br>1b - Inverting is enabled. |
| **2**<br>**INV2EN** | RW | Pair Channels 2 Inverting Enable |
| **1**<br>**INV1EN** | RW | Pair Channels 1 Inverting Enable |
| **0**<br>**INV0EN** | RW | Pair Channels 0 Inverting Enable |

### 49.6.24 FTU_SWOCTRL (FTU Software Output Control Register)

Offset : 94h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CHNOCV | | | | | | | | CHNOC | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_SWOCTRL field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **15-8**<br>**CHNOCV** | RW | Channel n Software Output Control Value<br>Bit(n) = 0b - The software output control forces 0 to the channel (n) output.<br>Bit(n) = 1b - The software output control forces 1 to the channel (n) output. |
| **7-0**<br>**CHNOC** | RW | Channel n Software Output Control Enable<br>Bit(n) = 0b - The channel (n) output is not affected by software output control.<br>Bit(n) = 1b - The channel (n) output is affected by software output control. |

### 49.6.25 FTU_PWMLOAD (FTU PWM Load Register)

Offset : 98h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | LDOK | 0 | CHNSEL | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

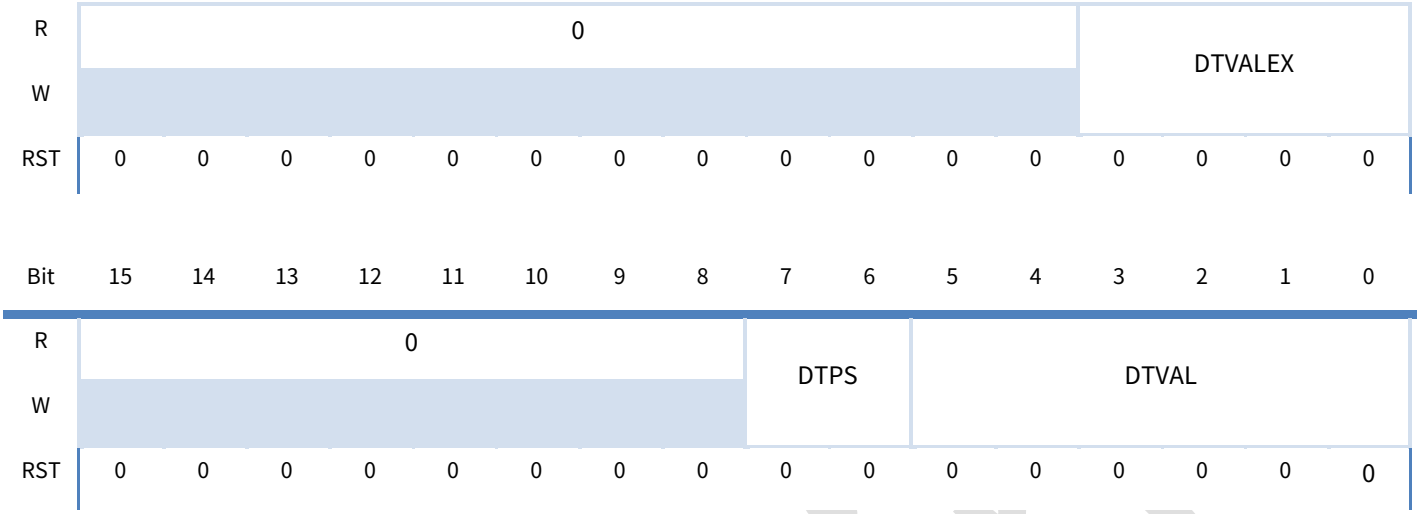
**FTU_PWMLOAD field descriptions**

| Field | Type | Description |
|---|---|---|
| **9**<br>**LDOK** | RW | Load Enable<br>Enables the loading of the MOD, CNTIN, and CV registers with the values of their buffers.<br>0b - Loading updated values is disabled.<br>1b - Loading updated values is enabled. |
| **7-0**<br>**CHNSEL** | RW | Channel n Select<br>0b - Channel match is not included as a reload opportunity.<br>1b - Channel match is included as a reload opportunity. |

### 49.6.26 FTU_PAIRnDEADTIME (Pair (n) Deadtime Configuration Register)

Offset : A0h+N*8h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|



| | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | | DTVALEX | | | |
| W | | | | | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | DTPS | | DTVAL | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FTU_PAIRnDEADTIME field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16**<br>**DTVALEX** | RW | Extended Deadtime Value<br>This field is a bit extension of the DTVAL field. The maximum deadtime value is extended to 1023 using the concatenation {DTVALEX, DTVAL}.<br>Deadtime insert value = (DTPS × {DTVALEX, DTVAL}). |
| **7-6**<br>**DTPS** | RW | Deadtime Prescaler<br>affect the deadtime counter clock frequency.<br>This field is write protected (depends on WPDIS/WPEN).<br>00b/01b - Divide the FTU input clock by 1.<br>10b - Divide the FTU input clock by 4.<br>11b - Divide the FTU input clock by 16. |
| **5-0**<br>**DTVAL** | RW | Deadtime Value<br>Selects the deadtime value.<br>This field is write protected (depends on WPDIS/WPEN). |

# Chapter 50 FC Programmable Interrupt Timer (FCPIT)

## 50.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

This chapter describes the FC Programmable Interrupt Timer (FCPIT) Module. It contains the following sections:

- Chip-specific FCPIT Information
- Introduction
- Features
- Functional Description
- Register Definition

## 50.2 Chip-specific Information

There is one FCPIT module with four channels in the FC7240F2M. Accessing the FCPIT when its functional clock is disabled may return a transfer error.

### 50.2.1 FCPIT/DMA Trigger

The periodic trigger output generated from the FCPIT will be sent to DMAMUX. The channel mapping can be found in the TRGSEL source table.

## 50.3 Overview

The FCPIT is a multi-channel timer module that can generate interrupts, trigger outputs independently when the timer reaches the programmed count. Different timer channels can operate individually or can be chained together by configuration. The FCPIT can operate in low power modes and can be used to wake up the chip. The trigger outputs can be used to trigger other modules on the device (i.e., DMA).

Other timer modules can also input triggers to the FCPIT to control the start of the timer channel.

FCPIT can work in compare mode for each timer channel.

- Compare Mode:
  The timer starts to decrease when enabled and generate trigger output when the timer counts to zero. The control registers decide when each timer channel will start, reload and restart. The timer can be configured to always decrease from a programmed start value. And the timer can start counting after enabling the timer, or being triggered by the selected input trigger, or being triggered by the previous channel timeout (when channels are chained). When enabling the channel chain function, the applications can achieve a larger timeout range.
- Capture Mode:
  The timer can be used to perform measurements to capture the timer value and store the inverse of the current counter value in the timer value register when a selected trigger input is asserted. The timer can support once-off or multiple measurements. For example, frequency measurements.

The timer channels operation clock is asynchronous with the register access clock. To ensure normal operations, clock synchronization is needed between different clock domains.

### 50.3.1 Modes of Operation

The following table describes the operation modes of the FCPIT module.

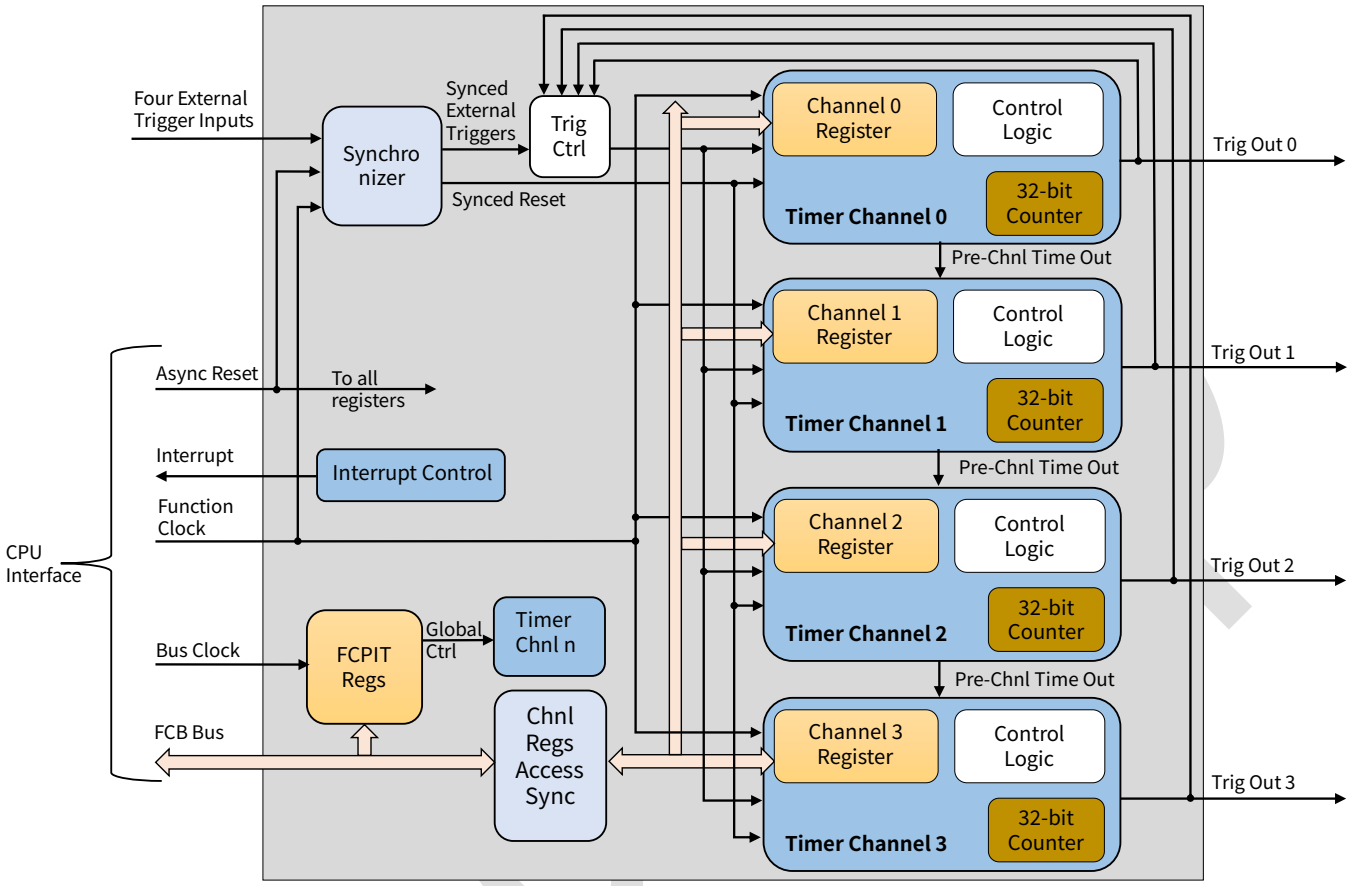
**Table 72. FCPIT modes of operation**

| Mode | Description |
|---|---|
| Run | The FCPIT operates normally. |
| Wait/Stop/ Standby | The FCPIT can continue to operate normally if the low power mode enable bit (PIT_MCR[LPM_EN]) is set and FCPIT is using a clock source that remains operating during low power modes. Low power wakeup is supported. |
| Debug | Can continue operating provided the Debug Enable bit (FCPIT_MCR[DBG_EN]) is set. |

## 50.4 Block Diagram

The following is the block diagram of the FCPIT module.

**Figure 118. FCPIT block diagram**




## 50.5 Features

The FCPIT module includes the following features:

- Four independent timer channels
- 32-bit programmable timeout value for each channel
- Support trigger output to other modules
- Support chaining the channels to get a larger timeout range
- Support capturing the current counter value
- Support external pulse to start the counter

## 50.6 Functional Description

### 50.6.1 Initialization

The steps to initialize the FCPIT module are as follows:

- Set the M_CEN bit in the FCPIT_MCR register to enable the peripheral clock.
- Need 4 peripheral clock cycles to wait for clock synchronization and reset de-assertion.
- Before enabling each timer channel, configure the timer operation mode (MODE bits), select the Trigger source (TRG_SEL & TRG_SRC) and Trigger control bits (TROT, TSOT, TSOI bits) in the FCPIT_TCTRLn register.
- To enable the channel chain function, configure the CHAIN bit in the FCPIT_TCTRLn register of the corresponding channel.
- When the channel working in Compare Mode, configure the FCPIT_TVAL register of appropriate value to set the start value for those channels.

- Configure TIEn bits in FCPIT_MIER register to enable the interrupt for corresponding channels to generate the interrupt when the timer timeout.
- Configure the low power mode of the module by setting the LPM_EN bits in the FCPIT_MCR register. This is the same for all timer channels.
- To enable the timer channels, set the T_EN bit in the FCPIT_TCRTLn register for the corresponding channel.
- When the channel working in Capture Mode, the captured timer value can be read from the FCPIT_TVALn register when channel timeout (detecting the rising edge of input trigger).
- At any time, the current value of each timer channel can be read from the corresponding FCPIT_CVALn register.
- The timer interrupt flag bits (TIFn) in the FCPIT_MSR register get asserted when timer timeout. These bits can be cleared by writing '1' to them.

### 50.6.2 Timer Modes

Three timer modes are supported by setting different values in the MODE bits in the FCPIT_TCTRLn register:

- **32-bit Periodic Counter**: In this mode, the counter will load the programmed value and then count down to zero. It will then set the timer interrupt flag and assert the output trigger.
- **Dual 16-bit Periodic Counter**: In this mode, the counter will load the programmed value and then the lower 16-bits will decrease down to zero, which will assert the output trigger. After then, the upper 16-bits start to decrease down to zero, which will deassert the output set the timer interrupt flag.
- **32-bit Trigger Accumulator**: In this mode, the counter will load the programmed value on the rising edge of the first input trigger and then start to decrease by 1 on the rising edge of each trigger until the counter counts to 0. It will then set the timer interrupt flag and assert the output trigger.
- **32-bit Trigger Input Capture**: In this mode, the counter will load with 0xFFFF_FFFF and then start to decrease. If a trigger rising edge is detected, it will capture the current counter value and then store the inverse of this value into the load value register. It will then set the timer interrupt flag and assert the output trigger.

The timer operation is further controlled by Trigger Control bits (TSOT, TSOI, TROT) which control the load, reload, start and restart of the timers.

### 50.6.3 Timers Control from Trigger

The TSOT, TROT, TSOI and TRG_SEL, TRG_SRC bits in the FCPIT_TCTRLn register control how the input trigger affects the timer operation. The TRG_SEL selects one input trigger for the channel from FCPIT internal triggers or external triggers. The TRG_SRC further selects which trigger source (internal trigger or external trigger) to input the channel.

The selected trigger will decide the timer operation by configuring TROT, TSOI & TSOT bits:

- If TSOI = 1, the counter stops on TIF assertion. To reload and decrease, need trigger rising (if TSOT = 1) or T_EN be set (if TSOT = 0). If TSOI = 0, the counter does not stop after timeout.
- If TROT = 1, the counter is loaded when each input trigger is asserted; otherwise, the counter is loaded whenever T_EN set or timeout rising edge (timeout not used in Capture modes).
- If TSOT = 1, the counter will start to decrease on the input trigger. Subsequent input triggers are ignored till a counter timeout. If TSOT = 0, the counter decreases on the next clock edge immediately. TSOT is useless when the channel is Chained .

In different timer modes, the bits above will have different functions:

- In 32-bit Periodic Counter and Dual 16-bit Periodic Counter modes, all bits (TSOT, TSOI & TROT) will control the timer operation as described above.
- In 32-bit Trigger Accumulator mode, only TSOI bit will control the timer function. TROT & TSOT bits have no effect on

timer operation.

- In 32-bit Input Trigger Capture mode, TSOI and TROT bits will control the timer function. TSOT bit has no effect on timer operation.

### 50.6.4 Channel-Chained Function

The channel chaining function can achieve a larger value of timeout by chaining the timer channels. This makes them work in a "nested loop" manner thereby leading to an effective timeout value of TVALCHn × (TVALCHn-1 + 1).

The channels are chained by setting the CHAIN bit in the corresponding channel's FCPIT_TCTRLCHn register. When a channel is chained, it counts one time on the previous channel timeout in all timer mode. The TSOT bit has no effect if a channel is chained.

## 50.7 Register Definition

FCPIT base address : 4002_E000h

**Register Overview - FCPIT**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **8h** | FCPIT_MCR (Module Control Register) | 32 | RW | 0000_0000h | page |
| **Ch** | FCPIT_MSR (Module Status Register) | 32 | W1C | 0000_0000h | page |
| **10h** | FCPIT_MIER (Module Interrupt Enable Register) | 32 | RW | 0000_0000h | page |
| **14h** | FCPIT_SETTEN (Set Timer Enable Register) | 32 | RW | 0000_0000h | page |
| **18h** | FCPIT_CLRTEN (Clear Timer Enable Register) | 32 | RW | 0000_0000h | page |
| **20h+n*10h** | FCPIT_TVALn (Timer Value Register) | 32 | RW | 0000_0000h | page |
| **24h+n*10h** | FCPIT_CVALn (Current Timer Value Register) | 32 | RO | FFFF_FFFFh | page |
| **28h+n*10h** | FCPIT_TCTRLn (Timer Control Register) | 32 | RW | 0000_0000h | page |

### 50.7.1 FCPIT_MCR (Module Control Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | DBG_EN | LPM_EN | SW_RST | M_CEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCPIT_MCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **3**<br>**DBG_EN** | RW | Debug Enable<br>Configure the timer channels to continue running or stop when the device enters the Debug mode<br>0b - Timer channels are stopped in Debug mode<br>1b - Timer channels continue running in Debug mode |
| **2**<br>**LPM_EN** | RW | Low Power Mode Enable<br>Configure the timer channels to continue running or stop when the device enters the LPM mode<br>0b - Timer channels are stopped in LPM mode<br>1b - Timer channels continue running in LPM mode |
| **1**<br>**SW_RST** | RW | Software Reset<br>Resets all channels and registers except the Module Control Register. Remains in the reset state until cleared by software.<br>To synchronize the clock and propagate the reset, software needs to wait for 4 peripheral clock cycles before clearing the Software Reset bit.<br>0b - Timer channels and registers are not reset<br>1b - Timer channels and registers are reset |
| **0**<br>**M_CEN** | RW | Module Clock Enable<br>This bit is used to enable the peripheral clock of FCPIT module. Must set this bit before writing/reading MSR, SETTEN, TCTRL, TVAL and CVAL registers. This bit also needs to be set before writing the CLRTEN register. To synchronize the clock and update register bits, the peripheral clock and bus clock must be enabled.<br>Reading/Writing the MSR, SETTEN, TCTRL, TVAL and CVAL registers while M_CEN = 0 will lead to assertion of transfer error for that bus cycle. If the peripheral clock is gated by other clock gating bits, need to configure those bits appropriately to enable the peripheral clock to this module.<br>0b - Disable peripheral clock of timers<br>1b - Enable peripheral clock of timers |

### 50.7.2 FCPIT_MSR (Module Status Register)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | TIF3 | TIF2 | TIF1 | TIF0 |
| W | | | | | | | | | | | | | w1c | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCPIT_MSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **3**<br>**TIF3** | W1C | Channel 3 Timer Interrupt Flag<br>In compare modes, set the flag to 1 at the end of the timer period. In capture modes, set the flag to 1 when the trigger asserts. Writing logic 1 to this flag will clear it. Writing 0 to it will have no effect.<br>0b - Timer has not timed out<br>1b - Timeout has occurred |
| **2**<br>**TIF2** | W1C | Channel 2 Timer Interrupt Flag<br>In compare modes, sets the flag to 1 at the end of the timer period. In capture modes, set the flag to 1 when the trigger asserts. Writing logic 1 to this flag will clear it. Writing 0 to it will have no effect.<br>0b - Timer has not timed out<br>1b - Timeout has occurred |
| **1**<br>**TIF1** | W1C | Channel 1 Timer Interrupt Flag<br>In compare modes, set the flag to 1 at the end of the timer period. In capture modes, set the flag to 1 when the trigger asserts. Writing logic 1 to this flag will clear it. Writing 0 to it will have no effect.<br>0b - Timer has not timed out<br>1b - Timeout has occurred |
| **0**<br>**TIF0** | W1C | Channel 0 Timer Interrupt Flag<br>In compare modes, sets the flag to 1 at the end of the timer period. In capture modes, sets the flag to 1 when the trigger asserts. Writing logic 1 to this flag will clear it. Writing 0 to it will have no effect.<br>0b - Timer has not timed out<br>1b - Timeout has occurred |

### 50.7.3 FCPIT_MIER (Module Interrupt Enable Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | TIE3 | TIE2 | TIE1 | TIE0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCPIT_MIER field descriptions**

| Field | Type | Description |
|---|---|---|
| **3**<br>**TIE3** | RW | Channel 3 Timer Interrupt Enable<br>Interrupt will be generated when this bit is set to 1 and corresponding Timer Interrupt Flag is asserted.<br>0b - Interrupt generation is disabled<br>1b - Interrupt generation is enabled |
| **2**<br>**TIE2** | RW | Channel 2 Timer Interrupt Enable<br>Interrupt will be generated when this bit is set to 1 and corresponding Timer Interrupt Flag is asserted.<br>0b - Interrupt generation is disabled<br>1b - Interrupt generation is enabled |
| **1**<br>**TIE1** | RW | Channel 1 Timer Interrupt Enable<br>Interrupt will be generated when this bit is set to 1 and corresponding Timer Interrupt Flag is asserted.<br>0b - Interrupt generation is disabled<br>1b - Interrupt generation is enabled |
| **0**<br>**TIE0** | RW | Channel 0 Timer Interrupt Enable<br>Interrupt will be generated when this bit is set to 1 and corresponding Timer Interrupt Flag is asserted.<br>0b - Interrupt generation is disabled<br>1b - Interrupt generation is enabled |

### 50.7.4 FCPIT_SETTEN (Set Timer Enable Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | SET_T_EN_3 | SET_T_EN_2 | SET_T_EN_1 | SET_T_EN_0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCPIT_SETTEN field descriptions**

| Field | Type | Description |
|---|---|---|
| **3**<br>**SET_T_EN_3** | RW | Enable Timer Channel 3<br>Writing "1" to this bit will enable the timer channel 3. Writing 0 will not disable the counter. To clear this bit, set T_EN bit in TCTRL3 to "0" or write "1" to the CLR_T_EN_3 bit in CLRTEN register.<br>Both this bit and T_EN bit in TCTRL3 register will enable the timer channel 3, but it is not necessary to set them at the same time.<br>0b - No effect<br>1b - Enables the Timer Channel 3 |
| **2**<br>**SET_T_EN_2** | RW | Enable Timer Channel 2<br>Writing "1" to this bit will enable the timer channel 2. Writing 0 will not disable the counter. To clear this bit, set T_EN bit in TCTRL2 to "0" or write "1" to the CLR_T_EN_2 bit in CLRTEN register.<br>Both this bit and T_EN bit in TCTRL2 register will enable the timer channel 2, but it is not necessary to set them at the same time.<br>0b - No effect<br>1b - Enables the Timer Channel 2 |
| **1**<br>**SET_T_EN_1** | RW | Enable Timer Channel 1<br>Writing "1" to this bit will enable the timer channel 1. Writing 0 will not disable the counter. To clear this bit, set T_EN bit in TCTRL1 to "0" or write "1" to the CLR_T_EN_1 bit in CLRTEN register.<br>Both this bit and T_EN bit in TCTRL1 register will enable the timer channel 1, but it is not necessary to set them at the same time.<br>0b - No effect<br>1b - Enables the Timer Channel 1 |
| **0**<br>**SET_T_EN_0** | RW | Enable Timer Channel 0<br>Writing "1" to this bit will enable the timer channel 0. Writing 0 will not disable the counter. To clear this bit, set T_EN bit in TCTRL0 to "0" or write "1" to the CLR_T_EN_0 bit in CLRTEN register.<br>Both this bit and T_EN bit in TCTRL0 register will enable the timer channel 0, but it is not necessary to set them at the same time.<br>0b - No effect<br>1b - Enables the Timer Channel 0 |

### 50.7.5 FCPIT_CLRTEN (Clear Timer Enable Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | CLR_T_EN_3 | CLR_T_EN_2 | CLR_T_EN_1 | CLR_T_EN_0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCPIT_CLRTEN field descriptions**

| Field | Type | Description |
|---|---|---|
| **3**<br>**CLR_T_EN_3** | RW | Clear Timer Channel 3 Enable<br>Writing a "1" to this bit will disable the timer channel 3 no matter if the channel is enabled by setting SET_T_EN_3 bit or T_EN bit in TCTRL3 register. This bit is self-clearing and will always read 0.<br>0b - No Action<br>1b - Clear T_EN bit for Timer Channel 3 |
| **2**<br>**CLR_T_EN_2** | RW | Clear Timer Channel 2 Enable<br>Writing a "1" to this bit will disable the timer channel 2 no matter if the channel is enabled by setting SET_T_EN_2 bit or T_EN bit in TCTRL2 register. This bit is self-clearing and will always read 0.<br>0b - No Action<br>1b - Clear T_EN bit for Timer Channel 2 |
| **1**<br>**CLR_T_EN_1** | RW | Clear Timer Channel 1 Enable<br>Writing a "1" to this bit will disable the timer channel 1 no matter if the channel is enabled by setting SET_T_EN_1 bit or T_EN bit in TCTRL1 register. This bit is self-clearing and will always read 0.<br>0b - No Action<br>1b - Clear T_EN bit for Timer Channel 1 |

| | | |
|---|---|---|
| **0**<br>**CLR_T_EN_0** | RW | Clear Timer Channel 0 Enable<br>Writing a "1" to this bit will disable the timer channel 0 no matter if the channel is enabled by setting SET_T_EN_0 bit or T_EN bit in TCTRL0 register. This bit is self-clearing and will always read 0.<br>0b - No action<br>1b - Clear T_EN bit for Timer Channel 0 |

### 50.7.6 FCPIT_TVALn (Timer Value Register)

Offset : 20h+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TMR_VAL | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TMR_VAL | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCPIT_TVALn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**TMR_VAL** | RW | Timer Value<br>In compare modes, sets the timer channel start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. A new value written to this register will not affect the current cycle and will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer channel must be disabled and enabled again.<br>In capture modes, the timer will count down from 0xFFFF_FFFF, and this register will store the inverse of the counter whenever the trigger asserts.<br>NOTE: 0000_0000h is invalid load value in compare modes.<br>0000_0001h - FFFF_FFFFh are the valid value to be loaded (Compare Mode) or value of Timer (Capture Mode) |

### 50.7.7 FCPIT_CVALn (Current Timer Value Register)

Offset : 24h+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TMR_CUR_VAL | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | TMR_CUR_VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**FCPIT_CVALn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**TMR_CUR_VAL** | RO | Current Timer Value<br>Read this register will return the current timer value when the timer is enabled. This register is read-only |

### 50.7.8 FCPIT_TCTRLn (Timer Control Register)

Offset : 28h+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | TRG_SEL | | | | TRG_SRC | 0 | | | | TROT | TSOI | TSOT |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | | | | | | | | | | | | MODE | | CHAIN | T_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCPIT_TCTRLn field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **27-24**<br>**TRG_SEL** | RW | Trigger Select<br>This field selects one trigger from 4 sources for starting and/or reloading the FCPIT timer. It should only be changed when the FCPIT timer channel is disabled. The TRG_SRC bit selects whether the internal trigger or the external trigger will be used for time channel.<br>The TRG_SEL bits select one trigger from the set of triggers (selected by TRG_SRC already) for each channel.<br>0000-0011b - Timer channel 0 - 3 trigger source is selected.<br>0100-1111b - Reserved. |
| **23**<br>**TRG_SRC** | RW | Trigger Source<br>Trigger Source includes internal or external trigger; this bit will select one type from them. The final trigger is selected by TRG_SEL and TRG_SRC. Refer to the chip configuration section for available external trigger options. If a channel does not have an associated external trigger, then this bit for that channel should be set to 1.<br>0b - Trigger source selected is external trigger<br>1b - Trigger source selected is internal trigger |
| **18**<br>**TROT** | RW | Timer Reload On Trigger<br>When set, the FCPIT timer will reload when the rising edge of the selected trigger input is detected.<br>The trigger input is ignored if the FCPIT is disabled during debug mode or lower power mode (LPM_EN or DBGEN = 0).<br>0b - Timer will not reload on the selected trigger<br>1b - Timer will reload on the selected trigger |
| **17**<br>**TSOI** | RW | Timer Stop On Interrupt<br>This bit controls whether the channel timer will stop counting after it times out.<br>0b - Timer continues to run and does not stop after timeout<br>1b - Timer will stop after timeout and will restart based on the TSOT configuration:<br>If TSOT = 1, the timer will restart when detecting the selected trigger asserted.<br>If TSOT = 0, the timer will restart when detecting the rising edge of T_EN (i.e., timer channel is disabled and then enabled). |
| **16**<br>**TSOT** | RW | Timer Start On Trigger<br>This bit controls when the timer starts counting down.<br>0b - Timer starts to decrease once the restart condition established (controlled by TSOI bit)<br>1b - Timer starts to decrease when the selected trigger asserted |

| | | |
|---|---|---|
| **3-2**<br>**MODE** | RW | Timer Operation Mode<br>Four operation modes for the timer. The timer decreases differently in these modes.<br>00b - 32-bit Periodic Counter<br>01b - Dual 16-bit Periodic Counter<br>10b - 32-bit Trigger Accumulator<br>11b - 32-bit Trigger Input Capture |
| **1**<br>**CHAIN** | RW | Chain Channel<br>When set, timer channel counts down once when channel N-1 timeout and asserting trigger. Channel 0 cannot be chained but will output the trigger to next channel when it is timeout.<br>0b - Disable Channel Chaining function. Channel Timer runs independently.<br>1b - Enable Channel Chaining function. Timer triggered by previous channel's timeout |
| **0**<br>**T_EN** | RW | Timer Enable<br>Enables or disables the Timer Channel<br>Two methods to disable the timer channel: set this bit to 0 or write 1 to CLR_T_EN_n of corresponding channel<br>0b - Disable Timer Channel<br>1b - Enable Timer Channel |

Flagchip

# Chapter 51 Timer Stamp (TSTMP)

## 51.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 51.2 Chip-specific Information

N/A

## 51.3 Features

- A 56-bit free-running timer counter
- 4 independent 32-bit width modulate timers with individual interrupts supported
- TSTMP0 counter clock from the SIRC1M clock, and TSTMP1 counter clock from the bus clock
- Support the selection of 4 modulate timer clock sources

## 51.4 Overview

The Timer Stamp (TSTMP) module has a 56-bit timer: TSTMP0, whose clock is from the SIRC1M clock, can be used to indicate the running time of the MCU system; TSTMP1 is used to indicate the CPU running time. The clock of TSTMP1 is from the bus clock.

## 51.5 Functional Description

The TSTMP can work in two counting modes for each modulate timer:

- Always Running Mode: the 4 independent modulate timer counters can generate interrupts with the MODx_SETVAL, MODx_ENABLE and MOD_INTEN set. To generate the next interrupt, the MODx SETVAL should be set with a value larger than the previous one because the modulate timer is always running. For example, if the MODx_SETVAL is first set n, then to generate the second interrupt, the MODx_SETVAL should be set n * (k + 1) (k = 1 for the second interrupt, k = 2 for the third one, and so on).
- Period Running Mode: the 4 independent modulate timer counters can generate interrupts with the MODx_SETVAL, MODx_ENABLE, MODx_MODE and MOD_INTEN set. When running to the MODx_SETVAL value, the timer will generate an interrupt and will be zero, then it will continue to run from zero. Whenever the timer is equal to the MODx_SETVAL, it will generate an interrupt and rerun from zero.

### 51.5.1 Modulate Timer Clock Source

When MODx_CLK is set, the modulate timer(0,1,2,x) clock source is the AON_CLK clock, otherwise, SIRC1M clock is selected for the modulate timer(0,1,2,x) as the clock source.

### 51.5.2 TSTMP Trigger

In standby mode, the TSTMP0 trigger as a wakeup source can be configured as a PWM output:

- Timerstamp trigger0: Set MOD0_MATCH and MOD1_MATCH (MOD1_MATCH value should be greater than MOD0_MATCH), the modulate timer1 will count until reaching the MOD0_MATCH value, then a posedge will occur; and a negedge will occur when the modulate timer1 reaches the MOD1_MATCH value.
- Timerstamp trigger1: Set MOD2_MATCH and MOD3_MATCH (MOD3_MATCH value should be greater than MOD2_MATCH), the modulate timer3 will count until reaching the MOD2_MATCH value, then a posedge will occur; and a negedge will occur when the modulate timer3 reaches the MOD3_MATCH value.

## 51.6 Register Definition

TSTMP0 base address : 4002_C000h

TSTMP1 base address : 4002_D000h

**Register Overview - TSTMP**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | TSTMP_VALL (Low Value Register) | 32 | RO | 0000_0000h | page |
| **4h** | TSTMP_VALH (High Value Register) | 32 | RO | 0000_0000h | page |
| **8h** | TSTMP_MOD_INTEN (Modulate Interrupt Enable Register) | 32 | RW | 0000_0000h | page |
| **Ch** | TSTMP_MOD_STATUS (Modulate Status Register) | 32 | W1C | 0000_0000h | page |
| **10h** | TSTMP_MOD0_SETVAL (Modulate0 Set Value Register) | 32 | RW | 0000_0000h | page |
| **14h** | TSTMP_MOD1_SETVAL (Modulate1 Set Value Register) | 32 | RW | 0000_0000h | page |
| **18h** | TSTMP_MOD2_SETVAL (Modulate2 Set Value Register) | 32 | RW | 0000_0000h | page |
| **1Ch** | TSTMP_MOD3_SETVAL (Modulate3 Set Value Register) | 32 | RW | 0000_0000h | page |

### 51.6.1 TSTMP_VALL (Low Value Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TVALL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TVALL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TSTMP_VALL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 TVALL** | RO | Low 32 bit of the 56-bit TSTMP timer value. |

### 51.6.2 TSTMP_VALH (High Value Register)

Note: The VALL register should be read before reading the VALH register.

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TVALH | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TVALH | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TSTMP_VALH field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-0**<br>**TVALH** | RO | High 24 bit of the 56-bit TSTMP timer value. |

### 51.6.3 TSTMP_MOD_INTEN (Modulate Interrupt Enable Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | MOD3_CLK | MOD2_CLK | MOD1_CLK | MOD0_CLK | 0 | | | | MOD3_ENABLE | MOD2_ENABLE | MOD1_ENABLE | MOD0_ENABLE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | MOD3_MODE | MOD2_MODE | MOD1_MODE | MOD0_MODE | 0 | | | | MOD3_INTEN | MOD2_INTEN | MOD1_INTEN | MOD0_INTEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TSTMP_MOD_INTEN field descriptions**

| Field | Type | Description |
|---|---|---|
| **27**<br>**MOD3_CLK** | RW | Modulate Timer Counter 3 Clock Source Select<br>0b - Select 1M clock<br>1b - Select AON_CLK clock |

| | | |
|---|---|---|
| **26**<br>**MOD2_CLK** | RW | Modulate Timer Counter 2 Clock Source Select<br>0b - Select 1M clock<br>1b - Select AON_CLK clock |
| **25**<br>**MOD1_CLK** | RW | Modulate Timer Counter 1 Clock Source Select<br>0b - Select 1M clock<br>1b - Select AON_CLK clock |
| **24**<br>**MOD0_CLK** | RW | Modulate Timer Counter 0 Clock Source Select<br>0b - Select 1M clock<br>1b - Select AON_CLK clock |
| **19**<br>**MOD3_ENABLE** | RW | Modulate Timer Counter 3 Enable<br>0b - Timer is not enabled<br>1b - Timer is enabled to count |
| **18**<br>**MOD2_ENABLE** | RW | Modulate Timer Counter 2 Enable<br>0b - Timer is not enabled<br>1b - Timer is enabled to count |
| **17**<br>**MOD1_ENABLE** | RW | Modulate Timer Counter 1 Enable<br>0b - Timer is not enabled<br>1b - Timer is enabled to count |
| **16**<br>**MOD0_ENABLE** | RW | Modulate Timer Counter 0 Enable<br>0b - Timer is not enabled<br>1b - Timer is enabled to count |
| **11**<br>**MOD3_MODE** | RW | Modulate Timer Counter 3 Counter Mode<br>0b - Timer is not cleared after the modulate counter matches the set value<br>1b - Timer is cleared after the modulate counter matches the set value |
| **10**<br>**MOD2_MODE** | RW | Modulate Timer Counter 2 Counter Mode<br>0b - Timer is not cleared after the modulate counter matches the set value<br>1b - Timer is cleared after the modulate counter matches the set value |
| **9**<br>**MOD1_MODE** | RW | Modulate Timer Counter 1 Counter Mode<br>0b - Timer is not cleared after the modulate counter matches the set value<br>1b - Timer is cleared after the modulate counter matches the set value |
| **8**<br>**MOD0_MODE** | RW | Modulate Timer Counter 0 Counter Mode<br>0b - Timer is not cleared after the modulate counter matches the set value<br>1b - Timer is cleared after the modulate counter matches the set value |

| | | |
|---|---|---|
| **3**<br>**MOD3_INTEN** | RW | Modulate Timer Counter 3 Interrupt Enable<br>0b - Disable modulate timer counter 3 interrupt<br>1b - Enable modulate timer counter 3 interrupt |
| **2**<br>**MOD2_INTEN** | RW | Modulate Timer Counter 2 Interrupt Enable<br>0b - Disable modulate timer counter 2 interrupt<br>1b - Enable modulate timer counter 2 interrupt |
| **1**<br>**MOD1_INTEN** | RW | Modulate Timer Counter 1 Interrupt Enable<br>0b - Disable modulate timer counter 1 interrupt<br>1b - Enable modulate timer counter 1 interrupt |
| **0**<br>**MOD0_INTEN** | RW | Modulate Timer Counter 0 Interrupt Enable<br>0b - Disable modulate timer counter 0 interrupt<br>1b - Enable modulate timer counter 0 interrupt |

### 51.6.4 TSTMP_MOD_STATUS (Modulate Status Register)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | MOD3_MATCH | MOD2_MATCH | MOD1_MATCH | MOD0_MATCH |
| W | | | | | | | | | | | | | w1c | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TSTMP_MOD_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **3**<br>**MOD3_MATCH** | W1C | Modulate Timer Counter 3 Match Flag<br>Writing 1 to this field can clear the flag.<br>0b - Modulate counter does not match the set value<br>1b - Modulate counter matches the set value |
| **2**<br>**MOD2_MATCH** | W1C | Modulate Timer Counter 2 Match Flag<br>Writing 1 to this field can clear the flag.<br>0b - Modulate counter does not match the set value<br>1b - Modulate counter matches the set value |

| | | |
|---|---|---|
| **1**<br>**MOD1_MATCH** | W1C | Modulate Timer Counter 1 Match Flag<br>Writing 1 to this field can clear the flag.<br>0b - Modulate counter does not match the set value<br>1b - Modulate counter matches the set value |
| **0**<br>**MOD0_MATCH** | W1C | Modulate Timer Counter 0 Match Flag<br>Writing 1 to this field can clear the flag.<br>0b - Modulate counter does not match the set value<br>1b - Modulate counter matches the set value |

### 51.6.5 TSTMP_MOD0_SETVAL (Modulate0 Set Value Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MOD0_SETVAL | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MOD0_SETVAL | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TSTMP_MOD0_SETVAL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**MOD0_SETVAL** | RW | Set Modulate Timer Counter 0 Value<br>Writing to this register will auto-start the modulate counter. |

### 51.6.6 TSTMP_MOD1_SETVAL (Modulate1 Set Value Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MOD1_SETVAL | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MOD1_SETVAL | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TSTMP_MOD1_SETVAL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**MOD1_SETVAL** | RW | Set Modulate Timer Counter 1 Value<br>Writing to this register will auto-start the modulate counter, but cannot be configured when TSTMP0 fly while. |

### 51.6.7 TSTMP_MOD2_SETVAL (Modulate2 Set Value Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MOD2_SETVAL | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MOD2_SETVAL | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TSTMP_MOD2_SETVAL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**MOD2_SETVAL** | RW | Set Modulate Timer Counter 2 Value<br>Writing to this register will auto-start the modulate counter, but cannot be configured when TSTMP0 fly while. |

### 51.6.8 TSTMP_MOD3_SETVAL (Modulate3 Set Value Register)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MOD3_SETVAL | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MOD3_SETVAL | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TSTMP_MOD3_SETVAL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**MOD3_SETVAL** | RW | Set Modulate Timer Counter 3 Value<br>Writing to this register will auto-start the modulate counter, but cannot be configured when TSTMP0 fly while. |

Flagchip

# Chapter 52 Real-Time Clock (RTC)

## 52.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 52.2 Chip-specific Information

N/A

## 52.3 Features

- A 32-bit seconds counter with roll-over protection that increments once every second
- A 16-bit prescaler that increments once every 32.768 kHz clock cycle with compensation
- Register write protection via corresponding LR bits, and unlock via Power-on Reset (POR), Low Voltage Detect Reset (LVR), or CR[SWRST]
- Select seconds interrupt frequency with 1, 2, 4, 8, 16, 32, 64, 128 Hz

## 52.4 Overview

- Interrupts from the Real-Time Clock (RTC) are enabled by default as described in the RTC_IER register
- RTC can be active in all low power modes and the time counter is clocked by default from 32.768 kHz and generate an interrupt to exit any low power mode.

## 52.5 Functional Description

### 52.5.1 Clock

The RTC's time counter can be clocked by default from a 32.768 kHz clock.

If the CR[CKPS]=0, the RTC_CLKOUT pin signal can select a square wave prescaler output (select 1, 2, 4, 8, 16, 32, 64, or 128 Hz).

If the CR[CKPS]=1, the RTC_CLKOUT pin signal can select the three different 32.768 kHz clocks and one FOSC_DIVL clock. The FOSC is 40 MHz and supports dividing with 1, 2, 4, 8, 16, 32 and 64, so the FOSC_DIVL could be 0.625 MHz, 1.25 MHz, 2.5 MHz, 5 MHz, 10 MHz, 20 MHz, and 40 MHz.

### 52.5.2 Time Counter

If it is necessary for software to read the Prescaler or Seconds register while RTC is enabled, two read accesses must be performed and software verifies that the same data was returned for both reads.

The Seconds register and Prescaler register can be written only when STR[TCE] is clear. Always write to the Prescaler register before writing to the Seconds register.

The Prescaler register increments provided STR[TCE] is set, STR[TOF] is clear, and the 32.768 kHz clock source is present.

If the Seconds register overflows, then STR[TOF] will be set and the Prescaler register will stop incrementing. Clear STR[TOF] by initializing the Seconds register. The Seconds register and Prescaler register read as zero whenever STR[TOF] is set.

The compensation logic can adjust the clock error by TCR[6:0]/(32768*(CIR[7:0]+1)) if TCR[7] is 0, or (128-TCR[6:0]/(32768*(CIR[7:0]+1) if TCR[7] is 1. If RTC_CLK is faster than the target 32.768K, set the TCR[7] to 1; otherwise, set TCR[7] to 0. The compensation factor must be calculated externally to the Compensation register. Updates to the Compensation register will not take effect until the next time the Seconds register increments and provided the previous compensation interval has expired. When the compensation interval is set to other than once a second, then the compensation is applied in the first second interval and the remaining second intervals receive no compensation. Compensation is disabled by configuring the time Compensation register to zero.

### 52.5.3 Interrupt

For alarm interrupts, the Alarm register (AR), STR[TAF], and IER[TAIE] allow the RTC to generate an alarm interrupt at a predefined time. The 32-bit TAR is compared with the 32-bit Seconds register (SR) each time it increments. STR[TAF] will be set when AR equals SR and SR increments. STR[TAF] is cleared by writing AR. Writing a value that is less than SR will prevent STR[TAF] from setting again. STR[TAF] can be cleared by writing to the AR register.

The RTC seconds interrupt is an edge-sensitive interrupt that is generated once a second. It is enabled in the RTC by the seconds interrupt enable bit. The frequency of the seconds interrupt defaults to 1 Hz, but can instead be configured to trigger every 2, 4, 8, 16, 32, 64, or 128 Hz.

### 52.5.4 Register Lock

The Lock register (LR) can be used to prevent write accesses to corresponding registers until the next POR or software reset. Locking the Control register (CR) will disable the software reset. Locking LR will block future updates to LR. Writing to a register protected by the lock register has no effect.

The CR[UM] configures software write access to the Time Counter Enable (STR[TCE]) field. When CR[UM] is clear, STR[TCE] can be written only when LR[STRL] is set. When CR[UM] is set, STR[TCE] can also be written when STR[TCE] is clear or when STR[TOF] is set. When LR[STRL] is set, CR[UM] does not affect STR[TCE].

The power-on-reset signal initializes all RTC registers to the default value. A software reset bit can also initialize all RTC registers. Set CR[SWRST] can force the equivalent of a POR to the rest of the RTC module. CR[SWRST] is not affected by the software reset and must be cleared by software.

## 52.6 Register Definition

RTC base address : 4003_0000h

**Register Overview - RTC**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | RTC_SR (Seconds Register) | 32 | RW | 0000_0000h | page |
| **4h** | RTC_PR (Prescaler Register) | 32 | RW | 0000_0000h | page |
| **8h** | RTC_AR (Alarm Register) | 32 | RW | 0000_0000h | page |
| **Ch** | RTC_COMPR (Compensation Register) | 32 | RW | 0000_0000h | page |
| **10h** | RTC_CR (Control Register) | 32 | RW | 0000_0000h | page |
| **14h** | RTC_STR (Status Register) | 32 | RW | 0000_0000h | page |
| **18h** | RTC_LR (Lock Register) | 32 | RW | 0000_0078h | page |
| **1Ch** | RTC_IER (Interrupt Enable Register) | 32 | RW | 0000_0006h | page |

### 52.6.1 RTC_SR (Seconds Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TSR | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSR | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RTC_SR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**TSR** | RW | Time Seconds Register<br>When set STR[TCE], the TSR is read only and increments, TSR will read as zero when STR[TOF] is set.<br>When clear STR[TCE], the TSR can be read or written. Writing to the TSR will clear the STR[TOF] . |

### 52.6.2 RTC_PR (Prescaler Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TPR | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RTC_PR field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**TPR** | RW | Time Prescaler Register<br>When set STR[TCE], the TPR is read-only and increments every 32.768 kHz clock cycle. TPR will read as zero when STR[TOF] is set.<br>When clear STR[TCE], the TPR can be read or written, SR[TSR] increments when TPR[14] transitions on the falling edge. |

### 52.6.3 RTC_AR (Alarm Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TAR | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TAR | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RTC_AR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**TAR** | RW | Time Alarm Register<br>When set STR[TCE], the STR[TAF] is set whenever the AR[TAR] equals the SR[TSR] and the SR[TSR] increments. |

### 52.6.4 RTC_COMPR (Compensation Register)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CIC | | | | | | | | TCV | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CIR | | | | | | | | TCR | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RTC_COMPR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-24**<br>**CIC** | RO | Compensation Interval Counter<br>Current value of the compensation interval counter. If the compensation interval counter equals zero, then it is loaded with the contents of the CIR. Reading this register at the same time as the seconds counter is incrementing can result in an incorrect value being read. |
| **23-16**<br>**TCV** | RO | Time Compensation Value<br>Current value used by the compensation logic for the second interval adjustment. Updated once a second if the CIC equals 0 with the contents of the TCR. If the CIC does not equal zero, then it is loaded with zero. Reading this register at the same time as the seconds counter is incrementing can result in an incorrect value being read. |
| **15-8**<br>**CIR** | RW | Compensation Interval Register<br>Configure the register to decide how many seconds to perform on TCR adjustment. This register is double buffered and writes do not take effect until the end of the current compensation interval. |
| **7-0**<br>**TCR** | RW | Time Compensation Register<br>Configure the register to decide how many prescale count adjust in one second. This register is double-buffered and writes do not take effect until the end of the current compensation interval.<br>TCR[7] is 0, prescaler overflows every 32768 - TCR[6:0] cycle<br>TCR[7] is 1, prescaler overflows every 32768 +128 - TCR[6:0] cycle |

### 52.6.5 RTC_CR (Control Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | CKO | 0 | | | CKPS | 0 | UM | 0 | | SWRST |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RTC_CR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **9**<br>**CKO** | RW | Clock Output Enable<br>0b - Enable the 32 kHz clock output to pad.<br>1b - Disable the 32 kHz clock output to pad. |
| **5**<br>**CKPS** | RW | Clock Output Pin Select<br>0b - RTC_CLKOUT is from the prescaler output clock selected by IER[TSIC].<br>1b - RTC_CLKOUT is from the 32.768 kHz clock. |
| **3**<br>**UM** | RW | Update Mode<br>Set CR[UM], the STR[TCE] can be written if the STR[TOF] is set or if the STR[TCE] is clear.<br>Set CR[UM] and STR[TCE], then STR[TCE] should only be written once STR[TOF] is set.<br>0b - Registers cannot be written when locked.<br>1b - Registers can be written when locked under limited conditions. |
| **0**<br>**SWRST** | RW | Software Reset<br>This bit is cleared by POR.<br>0b - Not any software reset event.<br>1b - Reset all RTC registers except for the CR[SWRST]. |

### 52.6.6 RTC_STR (Status Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | TCE | 0 | TAF | TOF | 0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**RTC_STR field descriptions**

| Field | Type | Description |
|---|---|---|
| **4**<br>**TCE** | RW | Time Counter Enable<br>0b - Disable time counter.<br>1b - Enable time counter. |

| | | |
|---|---|---|
| **2**<br>**TAF** | RO | Time Alarm Flag<br>This bit is cleared by writing the AR register.<br>0b - Time alarm has not occurred.<br>1b - Time alarm has occurred and the AR[TAR] equals the SR[TSR]. |
| **1**<br>**TOF** | RO | Time Overflow Flag<br>This bit is cleared by writing the SR register when TCE is cleared.<br>0b - Time overflow has not occurred.<br>1b - Time overflow has occurred and the SR and PR is read as zero. |

### 52.6.7 RTC_LR (Lock Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | LRL | STRL | CRL | CPL | 0 | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

**RTC_LR field descriptions**

| Field | Type | Description |
|---|---|---|
| **6**<br>**LRL** | RW | Lock Register Lock<br>This bit can be set only by POR or RTC software reset after being cleared.<br>0b - Lock Register is locked and writes fail.<br>1b - Lock Register is not locked and writes normal. |
| **5**<br>**STRL** | RW | Status Register Lock<br>This bit can be set only by POR or RTC software reset after being cleared.<br>0b - Status Register is locked and writes fail.<br>1b - Status Register is not locked and writes normal. |
| **4**<br>**CRL** | RW | Control Register Lock<br>This bit can be set only by POR or RTC software reset after being cleared.<br>0b - Control Register is locked and writes fail.<br>1b - Control Register is not locked and writes normal. |

| | | |
|---|---|---|
| **3**<br>**CPL** | RW | Compensation Register Lock<br>This bit can be set only by POR or RTC software reset after being cleared.<br>0b - Compensation Register is locked and writes fail.<br>1b - Compensation Register is not locked and writes normal. |

### 52.6.8 RTC_IER (Interrupt Enable Register)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | TSIC | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | TSIE | 0 | TAIE | TOIE | 0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

**RTC_IER field descriptions**

| Field | Type | Description |
|---|---|---|
| **18-16**<br>**TSIC** | RW | Timer Seconds Interrupt Configuration<br>Select the RTC Seconds interrupt and the RTC_CLKOUT prescale output frequency. This field should only be altered when TSIE is clear.<br>000b - 1 Hz.<br>001b - 2 Hz.<br>010b - 4 Hz.<br>011b - 8 Hz.<br>100b - 16 Hz.<br>101b - 32 Hz.<br>110b - 64 Hz.<br>111b - 128 Hz. |
| **4**<br>**TSIE** | RW | Time Seconds Interrupt Enable.<br>0b - Disable seconds interrupt.<br>1b - Enable seconds interrupt. |
| **2**<br>**TAIE** | RW | Time Alarm Interrupt Enable<br>0b - Disable time alarm interrupt.<br>1b - Enable time alarm interrupt. |

| | | |
|---|---|---|
| **1**<br>**TOIE** | RW | Time Overflow Interrupt Enable<br>0b - Disable time overflow interrupt.<br>1b - Enable time overflow interrupt. |

Flagchip

# Chapter 53 Always-on Timer (AONTIMER)

## 53.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 53.2 Chip-specific Information

N/A

## 53.3 Features

- Support a 1-channel 16-bit timer counter or pulse counter with compare.
- Support asynchronous interrupt requests to wake up from any low power mode.
- Support hardware trigger assertion.
- Counter support overflow reset or free-running mode.
- Selectable functional clock for a prescaler or glitch filter.
- Selectable input source for pulse counter.

## 53.4 Overview

The Always-on Timer (AONTIMER) can increment normally as a timer counter with an optional prescaler, or as a pulse counter with an optional glitch filter in all power modes. It can also continue operating through most system reset events, allowing it to be used as a time of day counter because the AONTIMER is reset only on global Power-on Reset (POR) or Low Voltage Detect (LVD).

## 53.5 Functional Description

### 53.5.1 AONTIMER Clocking

The AONTIMER prescaler/glitch filter can be clocked by the input clock selected by PSR[PCS].

In pulse counter mode with the PSR[PBYP] set, the CNR increments directly by the input source selected by CSR[TPS] without other clock sources. In this case, configure the prescaler input clock source for a clock that is not toggling to minimize power.

**Table 73. Pulse counter input options**

| AONTIMER_CSR[TPS] | Pulse Counter Input Number | Chip Input |
|---|---|---|
| 00 | 0 | AONTIMER_CLK0 pin |
| 01 | 1 | AONTIMER_CLK1 pin |
| 10 | 2 | AONTIMER_CLK2 pin |
| 11 | 3 | TRGSEL output |

### 53.5.2 AONTIMER Prescaler

In Timer Counter mode, when PBYP is clear, the CNR is clocked by the prescaler output clock and increments every 2^2 to 2^16 prescaler clock cycles.

In Timer Counter mode, When PBYP is set, the CNR increments directly by the prescaler clock selected by PSR[PCS]. When the AONTIMER is enabled, the first increment of CNR takes an additional one or two prescaler clock cycles due to synchronization logic.

### 53.5.3 AONTIMER Glitch Filter

In Pulse Counter mode, when PBYP is clear, the CNR increments by the glitch filter output clock directly. When the AONTIMER is first enabled, the output of the glitch filter is asserted, that is, logic 1 for active-high and logic 0 for active-low. If the selected input source remains deasserted for at least 2^1 to 2^15 consecutive prescaler clock rising edges, then the glitch filter output also deasserts. If the selected input source remains asserted for at least 2^1 to 2^15 consecutive prescaler clock rising-edges, then the glitch filter output also asserts. The CNR increments each time the glitch filter output asserts. In Pulse Counter mode, the maximum rate at which the CNR can increment is once every 2^2 to 2^16 prescaler clock edges. When first enabled, the glitch filter waits for an additional one or two prescaler clock edges due to synchronization logic.

In Pulse Counter mode, when PBYP is set, the CNR increments directly once the selected input source asserts. Before the AONTIMER is first enabled, the selected input source is forced to be asserted. This prevents the CNR from incrementing if the selected input source is already asserted when the AONTIMER is first enabled.

### 53.5.4 AONTIMER Counter

The CNR increments by one on the following condition:

- Every prescaler clock with prescaler bypassed in timer counter mode
- Every prescaler output with prescaler enabled in timer counter
- Every input source assertion with glitch filter bypassed in pulse counter mode
- Every glitch filter output with glitch filter enabled in pulse counter mode.

The CNR is reset when the AONTIMER is disabled or the counter register overflows. If CSR[TFC] is cleared, then the CNR is also reset whenever CSR[TCF] is set.

The CNR cannot be initialized but can be read at any time. On each read of the CNR, software must first write to the CNR with any value. This synchronizes and registers the current value of the CNR into a temporary register. The contents of the temporary register are returned on each read of the CNR.

In Debug mode, the CNR increments normally in pulse counter mode, but stops incrementing in timer counter mode.

### 53.5.5 AONTIMER Compare

When the CNR matches the CMR value and increments, the following events occur: CSR[TCF] is set, interrupt requests are asserted with CSR[TIE] set, DMA requests are asserted with CSR[TDRE] set, hardware triggers are asserted, and CNR is reset with CSR[TFC] cleared.

When the AONTIMER is enabled, the CMR can be altered only when CSR[TCF] is set. When updating the CMR, the CMR must be written and CSR[TCF] must be cleared before the AONTIMER counter has incremented past the new AONTIMER compare value.

### 53.5.6 AONTIMER Interrupt

The interrupt asserts whenever CSR[TIE] and CSR[TCF] are set. CSR[TCF] is cleared by clear CSR[TEN] or by writing a logic 1 to it. CSR[TIE] can be altered and CSR[TCF] can be cleared while the AONTIMER is enabled. The interrupt is generated asynchronously to the system clock and can be used as a wake-up source to exit any low-power mode.

### 53.5.7 AONTIMER DMA Request

When TDRE is set, the AONTIMER DMA Request is generated whenever TCF is also set and the TCF is cleared until the DMA Controller is done.

### 53.5.8 AONTIMER Hardware Trigger

The AONTIMER hardware trigger asserts at the same time the CSR[TCF] is set and can be used to trigger hardware events in other peripherals without software intervention. The hardware trigger is always enabled. When the CMR is set to 0 with CSR[TFC] clear, then the hardware trigger asserts on the first compare and does not deassert. When the CMR is set to a nonzero value or if CSR[TFC] is set, the hardware trigger asserts on each compare and deasserts on the following increment of the CNR.

## 53.6 Register Definition

AONTIMER base address : 4002_F000h

**Register Overview - AONTIMER**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | AONTIMER_CSR (Control Status Register) | 32 | RW | 0000_0000h | page |
| **4h** | AONTIMER_PSR (Prescale Register) | 32 | RW | 0000_0000h | page |
| **8h** | AONTIMER_CMR (Compare Register) | 32 | RW | 0000_0000h | page |
| **Ch** | AONTIMER_CNR (Counter Register) | 32 | RW | 0000_0000h | page |

### 53.6.1 AONTIMER_CSR (Control Status Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | DBGEN | TDRE | TCF | TIE | TPS | | TPP | TFC | TMS | TEN |
| W | | | | | | | | | w1c | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AONTIMER_CSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **9**<br>**DBGEN** | RW | Debug Mode Enable<br>0b - Disable counter in debug mode<br>1b - Enable counter in debug mode |
| **8**<br>**TDRE** | RW | Timer DMA Request Enable<br>0b - Disable timer DMA request when CSR[TCF] is set.<br>1b - Enable timer DMA request when CSR[TCF] is set. |
| **7**<br>**TCF** | W1C | Timer Compare Flag<br>TCF is cleared by clear CSR[TEN] or write a logic 1 to it.<br>0b - CNR value is not equal to CMR and increments.<br>1b - CNR value is equal to CMR and increments. |

| | | |
|---|---|---|
| **6**<br>**TIE** | RW | Timer Interrupt Enable<br>0b - Disable timer interrupt when CSR[TCF] is set.<br>1b - Enable timer interrupt when CSR[TCF] is set. |
| **5-4**<br>**TPS** | RW | Timer Pin Select<br>Select the input source for Pulse Counter mode. TPS must be altered only when CSR[TEN] cleared.<br>00b - Select AONTIMER_CLK0 pin for Pulse Counter mode.<br>01b - Select AONTIMER_CLK1 pin for Pulse Counter mode.<br>10b - Select AONTIMER_CLK2 pin for Pulse Counter mode.<br>11b - Select TRGSEL output for Pulse Counter mode. |
| **3**<br>**TPP** | RW | Timer Pin Polarity<br>Select the polarity of the input source in Pulse Counter mode. TPP must be altered only when CSR[TEN] cleared<br>0b - Pulse Counter input source is active-high, and the CNR increments on the rising-edge.<br>1b - Pulse Counter input source is active-low, and the CNR increments on the falling-edge. |
| **2**<br>**TFC** | RW | Timer Free-Running Counter<br>TFC must be altered only when CSR[TEN] cleared.<br>0b - CNR is reset whenever TCF is set.<br>1b - CNR is reset on overflow. |
| **1**<br>**TMS** | RW | Timer Mode Select<br>Select timer mode. TMS must be altered only when CSR[TEN] cleared.<br>0b - Timer Counter mode.<br>1b - Pulse Counter mode. |
| **0**<br>**TEN** | RW | Timer Enable<br>0b - Disable the timer and reset internal logic, including the CNR and TCF.<br>1b - Enable the timer. |

### 53.6.2 AONTIMER_PSR (Prescale Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | PRESCALE | | | | PBYP | PCS | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AONTIMER_PSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **6-3**<br>**PRESCALE** | RW | Prescale Value<br>PRESCALE must be altered only when CSR[TEN] cleared.<br>In timer counter mode, the prescaler clock size is 2^(PRESCALE+1).<br>In pulse counter mode, the glitch filter recognizes change on input pin after 2^PRESCALE rising clock edges, not support PRESCALE is 0. |
| **2**<br>**PBYP** | RW | Prescaler Bypass<br>PBYP must be altered only when CSR[TEN] cleared.<br>0b - Enable the prescaler in timer counter mode or glitch filter in pulse mode.<br>1b - Bypass the prescaler in timer counter mode or glitch filter in pulse mode. |
| **1-0**<br>**PCS** | RW | Prescaler Clock Select<br>Selects the input clock for the prescaler in timer counter mode or for the glitch filter in pulse counter mode. PCS must be altered only when CSR[TEN] cleared.<br>00b - Selects the input clock 0 for the prescaler or glitch filter.<br>01b - Reserved.<br>10b - Selects the input clock 1 for the prescaler or glitch filter.<br>11b - Selects the input clock 2 for the prescaler or glitch filter.<br>Note: For details on the input clock selection, refer to Section 9.2.15 AONTIMER |

### 53.6.3 AONTIMER_CMR (Compare Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | COMPARE | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AONTIMER_CMR field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**COMPARE** | RW | Compare Value<br>When CSR[TEN] is set and CNR matches the CMR value and increments, TCF is set and the hardware trigger asserts until the next time the CNR increments. If the CMR is 0, the hardware trigger remains asserted until the CSR[TEN] cleared. If the CSR[TEN] is set, the CMR must be altered only when TCF is set. |

### 53.6.4 AONTIMER_CNR (Counter Register)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COUNTER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AONTIMER_CNR field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**COUNTER** | RW | Counter Value<br>The CNR returns the current value of the AONTIMER counter at the time this register was last written. |

# Chapter 54 Programmable Timer (PTIMER)

## 54.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | Added the bit field description to the register sections *PTIMER_DLY4_CHn, PTIMER_DLY5_CHn, PTIMER_DLY6_CHn,* and *PTIMER_DLY7_CHn* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 54.2 Chip-specific Information

N/A

## 54.3 Features

The Programmable Timer (PTIMER) has the below features:

- Up to 1 software trigger source
- Up to 2 hardware trigger sources, including from TRGSEL and Analog-to-Digital Converter (ADC) Conversion Complete (COCO) output
- One PTIMER cooperates with one ADC
- Up to 4 PTIMER channels and 8 programmable delays for each channel
- 1 channel corresponds to 1 ADC trigger input; 1 programmable delay corresponds to 1 ADC pre-trigger input
- One 16-bit programmable delay per pre-trigger output
- Pre-trigger outputs can be enabled or disabled independently
- Optional Back-to-Back (BTB) Mode operation, which enables the ADC conversions complete to trigger the next PTIMER channel
- Optional bypass of the delay registers of the pre-trigger outputs
- Up to 1 pulse output with programmable pulse delay and width
- Optional operation in One-Shot or Continuous Modes
- One channel flag and one sequence error flag per pre-trigger
- One programmable delay interrupt
- One sequence error interrupt
- Direct Memory Access (DMA) support

## 54.4 Overview

### 54.4.1 Instantiation Information

The FC7240F2M supports 2 PTIMERs.

### 54.4.2 PTIMER Trigger Output Connections with ADC and TRGSEL

Refer to the ADC trigger source section for PTIMER-ADC triggering scheme.

When PTIMER channel triggers work as trigger sources of the TRGSEL module, the PTIMER can be used to trigger other modules besides ADC.

There are 5 trigger outputs for PTIMER, including triggers from 4 channels and 1 from pulse-out.

### 54.4.3 PTIMER Software Trigger Source

The PTIMER supports software triggers. The PTIMER can be triggered by software through STATUS_CTRL[SWTRG].

### 54.4.4 PTIMER Trigger Input from TRGSEL

There is only one trigger input from TRGSEL for each PTIMER: PTIMER_TRG0. The PTIMER supports different trigger sources through the TRGSEL input side.

### 54.4.5 Back-to-back Operation Mode with ADC

With Back-to-back operation mode, PTIMER can trigger the next channel pre-trigger and trigger output based on the ADC COCO flag.

Flagchip

## 54.5 Functional Description

### 54.5.1 PTIMER Pre-trigger and Trigger Outputs

#### *54.5.1.1 PTIMER Counter Start*

The PTIMER contains a counter whose output is compared to several different digital values. If the PTIMER is enabled, then a trigger input event will reset the counter and make it start to count. A trigger input event is defined as a rising edge being detected on a selected trigger input source, or if a software trigger is selected and the Software Trigger bit (STATUS_CTRL[SWTRG]) is written with 1.

#### *54.5.1.2 Delay Calculation*

For each channel, a delay m determines the time between the assertion of the trigger input event to the time at which changes in the pre-trigger m output signal are started. The time is defined as:

- Trigger input event to pre-trigger m = (pre-scaler × multiplication factor × delay m) + 2 peripheral clock cycles
- Add 1 additional peripheral clock cycle to determine the time when the channel trigger output changes.

#### *54.5.1.3 Pre-trigger Bypass*

The delay in the channel delay register (DLYm_CHn) can be optionally bypassed, if the PTIMER Channel Pre-Trigger Output Select (CTRL_CHn[PTOS[m]]) is cleared. In this case, when the trigger input event occurs, the pre-trigger m is asserted after 2 peripheral clock cycles.

### 54.5.2 PTIMER Triggering ADC

#### *54.5.2.1 ADC Precondition by Pre-trigger*

Each PTIMER is associated with 1 ADC block. The PTIMER channel n (n < N, N = 4) pre-trigger outputs 0 to M-1 (M = 8); each pre-trigger output is connected to ADC pre-trigger inputs. The pre-triggers are used to precondition the ADC block before the actual trigger occurs. When the ADC receives the rising edge of the trigger, the ADC will start the conversion according to the precondition determined by the pre-triggers. The ADC contains N × M sets of configurations (ADC_SC[i], i = n × 8 + m) and result (ADC_RESULT[i], i = n × 8 + m) registers, allowing it to alternate conversions between different analog sources (refer to the ADC chapter for the available analog sources). The pre-trigger outputs are used to specify the signal from which the ADC channel will be sampled next. When a pre-trigger m is asserted, the ADC conversion is triggered with set i of the configuration (ADC_SC[i], i = n × 8 + m) and result registers (ADC_RESULT[i], i = n × 8 + m).

#### *54.5.2.2 Back-to-Back Operation*

The PTIMER can be configured for back-to-back operation. Back-to-back operation enables the ADC conversion completions to trigger the next PTIMER channel pre-trigger and trigger outputs, so that the ADC conversions can be triggered on the next set of configuration and results registers. When a back-to-back operation is enabled by setting the PTIMER Channel Pre-Trigger Back-to-Back Operation Enable (CTRL_CHn[BTB[m]]), then the delay m is ignored, and the pre-trigger m is asserted 2 peripheral cycles after the acknowledgement m is received.

The PTIMER back-to-back operation is supported on all channels of each PTIMER. The following PTIMER back-to-back operation sequence connections are configured from SCM_ADC_ROUTING[PTIMER01LOOP], SCM_ADC_ROUTING[PTIMER01LOOP_1] and SCM_ADC_ROUTING[PTIMER01LOOP_2]. The PTIMER back-to-back forming a ring is supported for channel 0 of two PTIMER and channel 0 and channel 1 cumulatively. The mappings of SCM_ADC_ROUTING to PTIMER loop mode are shown in the table below:

Flagchip

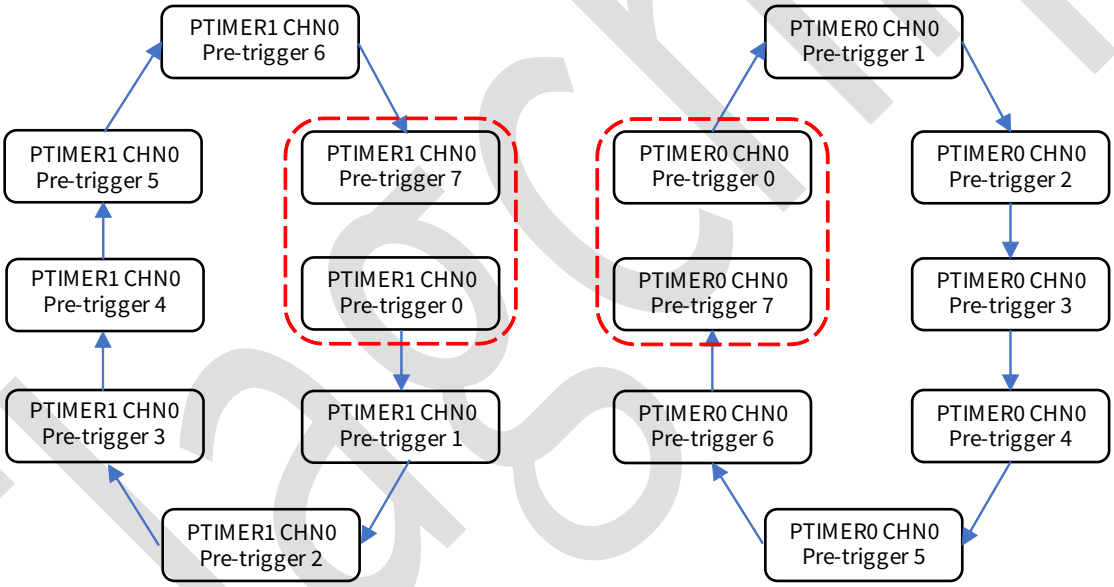
**Table 74. PTIMER loop mode**

| Mode | PTIMER01LOOP | PTIMER01LOOP_1 | PTIMER01LOOP_2 | PTIMER Loop Mode |
|---|---|---|---|---|
| **0** | 0 | 0 | 0 | PTIMER0 CH0 loop<br>PTIMER1 CH0 loop |
| **1** | x | 0 | 1 | PTIMER1 CH0 and CH1 in loop mode |
| **2** | x | 1 | 0 | PTIMER0 CH0 and CH1 in loop mode |
| **3** | x | 1 | 1 | PTIMER1 CH0 and CH1 in loop mode<br>PTIMER0 CH0 and CH1 in loop mode |
| **4** | 1 | 0 | 0 | PTIMER0 CH0 and PTIMER1 CH0 in loop mode |

For mode 0, the 8 pre-triggers inside CH0 form a ring by back-to-back operation of

- PTIMER0 CH0 pre-trigger 7 to PTIMER0 CH0 pre-trigger 0
- PTIMER1 CH0 pre-trigger 7 to PTIMER1 CH0 pre-trigger 0

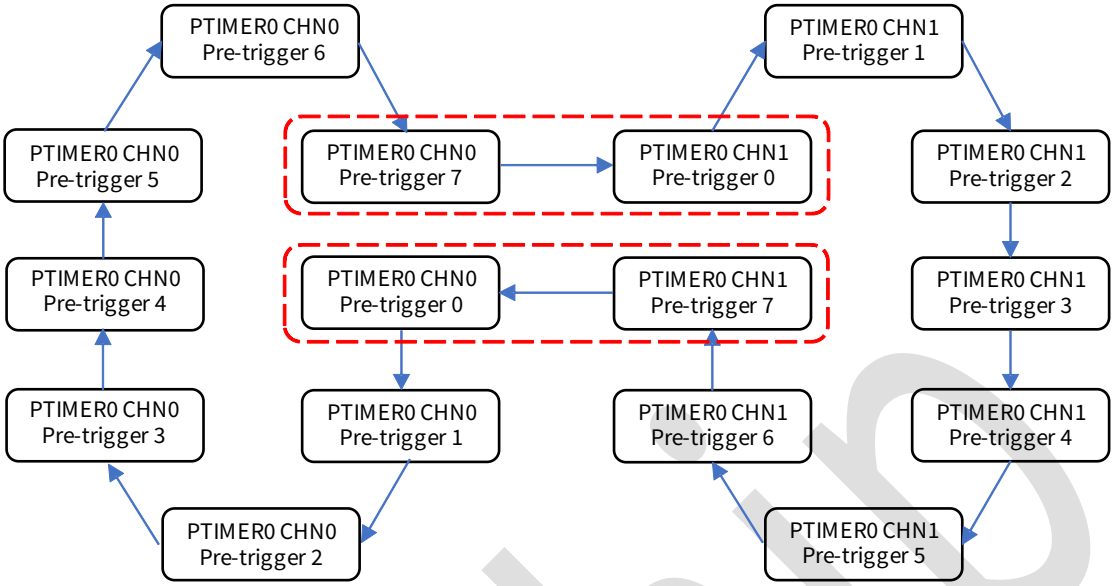
**Figure 119. PTIMER back-to-back operation in mode 0**




For mode 2, the cross PTIMER ring is implemented by back-to-back operation of

- PTIMER0 CH0 pre-trigger 7 to PTIMER0 CH1 pre-trigger 0
- PTIMER0 CH1 pre-trigger 7 to PTIMER0 CH0 pre-trigger 0

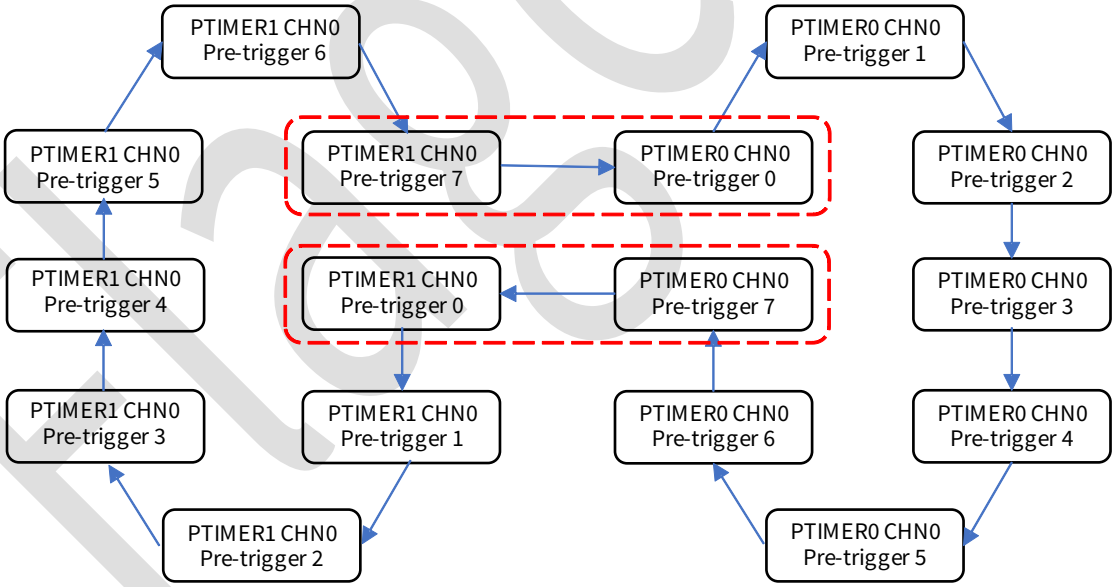
**Figure 120. PTIMER back-to-back operation in mode 2**




For mode 4, the cross PTIMER ring is implemented by back-to-back operation of

- PTIMER1 CH0 pre-trigger 7 to PTIMER0 CH0 pre-trigger 0
- PTIMER0 CH0 pre-trigger 7 to PTIMER1 CH0 pre-trigger 0

**Figure 121. PTIMER back-to-back operation in mode 4**




#### *54.5.2.3 Pre-trigger Active Lock*

When a pre-trigger m from a PTIMER channel n is asserted, the associated lock of the pre-trigger becomes active. The associated lock is released by the rising edge of the corresponding ADC_SC[i][COCO] (i = 8 × n + m). The ADC_SC[i][COCO] (i = 8 × n + m) is automatically cleared after the conversion result from ADC_RESULT[i] (i = 8 × n + m) is read. By either clearing ADC_SC[i][COCO] (i = 8 × n + m) or reading result from ADC_RESULT[i] (i = 8 × n + m), the next rising edge of ADC_SC[i][COCO] (i = 8 × n + m) can be generated to clear the lock later. The lock becomes inactive when the rising edge of corresponding ADC_SC[i][COCO] (i = 8 × n + m) occurs, the corresponding PTIMER pre-trigger is disabled, or the PTIMER is disabled.

The channel n trigger output is suppressed when any of the locks of the pre-triggers in channel n is active. The pre-trigger active lock works inside a single channel. Cross channel pre-trigger lock is not supported.

### 54.5.3 PTIMER Functions

#### *54.5.3.1 Pulse Output Configuration*

PTIMER can generate pulse outputs of configurable delay and width.

- When the PTIMER counter reaches the value set in PODLY[DLY1], then the Pulse-Out goes high.
- When the PTIMER counter reaches PODLY[DLY2], then it goes low.

PODLY[DLY2] can be set either greater or less than PODLY[DLY1].

ADC pre-trigger/trigger outputs and Pulse-Out generation have the same time base because they both share the PTIMER counter.

#### *54.5.3.2 Timing-related Registers Update Scheme*

Timing-related registers may need to become effective at the same time in some of the applications. There are buffer registers implemented for these timing-related registers in PTIMER. Any values written to them are written to their buffers first. The following registers follow the update scheme selected from STATUS_CTRL[LDMODE].

- MAX_CNT, PTIMER maximum counter register
- INT_DLY, PTIMER Interrupt Delay register
- DLYm_CHn, PTIMER Channel n Delay m register
- PODLY, PTIMER Pulse-Out Delay register

If STATUS_CTRL[LDMODE]=00, the internal registers are loaded with the values from their buffers immediately after 1 is written to STATUS_CTRL[LDOK].

If STATUS_CTRL[LDMODE]=01, the internal registers are loaded with the values when the PTIMER counter reaches PTIMER_MAX_CNT[MAX_CNT] + 1 value, after 1 is written to STATUS_CTRL[LDOK].

If STATUS_CTRL[LDMODE]=10, the internal registers are loaded with the values when a trigger input event is detected after 1 is written to STATUS_CTRL[LDOK].

If STATUS_CTRL[LDMODE]=11, the internal registers are loaded with the values either the PTIMER counter reaches PTIMER_MAX_CNT[MAX_CNT] + 1 value or a trigger input event is detected, after 1 is written to STATUS_CTRL[LDOK].

The buffers cannot be written until the values in buffers are loaded into their internal registers, after 1 is written to STATUS_CTRL[LDOK]. The application code can read STATUS_CTRL[LDOK] to determine the updates to the internal registers, as STATUS_CTRL[LDOK] is self-cleared when the internal registers are loaded.

#### *54.5.3.3 Interrupt and DMA Requests*

The interrupt flag STATUS_CTRL[INTFLAG] is set when the PTIMER counter reaches the value INT_DLY[INT_DLY]. An interrupt can be generated if the PTIMER Interrupt Enable (STATUS_CTRL[INTEN]) is set and the DMA Enable (STATUS_CTRL[DMAEN]) is cleared. If the DMA Enable (STATUS_CTRL[DMAEN]) is set, then the PTIMER requests a DMA transfer when the PTIMER Interrupt Flag (STATUS_CTRL[INTFLAG]) is set.

#### *54.5.3.4 Continuous Mode*

The value in the MAX_CNT register is the maximum value the counter can reach. It is used to reset the counter back to zero at the end of the count. If the Continuous Mode Enable (STATUS_CTRL[CONT]) is 0, the counter operation will stop until the next trigger input event occurs. Otherwise, the counter will resume a new count until PTIMER is disabled.

#### *54.5.3.5 PTIMER Error Mechanism*

A PTIMER Channel Sequence Error Flag STATUS_CHn[SERR_FLAG[m]] (from pre-trigger m) is set if a new pre-trigger m asserts when there is an active lock in the PTIMER channel n. If the PTIMER Sequence Error Interrupt Enable STATUS_CTRL[SERR_INTEN] is set, then the sequence error interrupt is generated.

The pre-trigger delay configured to 0 cycles in the same channel will assert a PTIMER channel sequence error. And ADC will not start a conversion.

For situations when PTIMER triggering is done through different PTIMER channels, software must ensure sufficient delays in between the pre-triggers, as the PTIMER reports channel sequence errors only for pre-triggers in the same PTIMER channel.

## 54.6 Register Definition

PTIMER0 base address : 4003_7000h

PTIMER1 base address : 4003_8000h

**Register Overview - PTIMER**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | PTIMER_STATUS_CTRL (Status and Control Register) | 32 | RW | 0000_0000h | page |
| **4h** | PTIMER_MAX_CNT (Max Count Number Register) | 32 | RW | 0000_FFFFh | page |
| **8h** | PTIMER_CNT (Counter Register) | 32 | RW | 0000_0000h | page |
| **Ch** | PTIMER_INT_DLY (Interrupt Delay Register) | 32 | RW | 0000_FFFFh | page |
| **10h+n*28h** | PTIMER_CTRL_CHn (Channel n Control Register) | 32 | RW | 0000_0000h | page |
| **14h+n*28h** | PTIMER_STATUS_CHn (Channel n Status Register) | 32 | RW | 0000_0000h | page |
| **18h+n*28h** | PTIMER_DLY0_CHn (Channel n Delay 0 Register) | 32 | RW | 0000_0000h | page |
| **1Ch+n*28h** | PTIMER_DLY1_CHn (Channel n Delay 1 Register) | 32 | RW | 0000_0000h | page |
| **20h+n*28h** | PTIMER_DLY2_CHn (Channel n Delay 2 Register) | 32 | RW | 0000_0000h | page |
| **24h+n*28h** | PTIMER_DLY3_CHn (Channel n Delay 3 Register) | 32 | RW | 0000_0000h | page |
| **28h+n*28h** | PTIMER_DLY4_CHn (Channel n Delay 4 Register) | 32 | RW | 0000_0000h | page |
| **2Ch+n*28h** | PTIMER_DLY5_CHn (Channel n Delay 5 Register) | 32 | RW | 0000_0000h | page |
| **30h+n*28h** | PTIMER_DLY6_CHn (Channel n Delay 6 Register) | 32 | RW | 0000_0000h | page |
| **34h+n*28h** | PTIMER_DLY7_CHn (Channel n Delay 7 Register) | 32 | RW | 0000_0000h | page |
| **190h** | PTIMER_POEN (Pulse-Out Enable Register) | 32 | RW | 0000_0000h | page |
| **194h** | PTIMER_PODLY (Pulse-Out Delay Register) | 32 | RW | 0000_0000h | page |

### 54.6.1 PTIMER_STATUS_CTRL (Status and Control Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | LDMODE | | SERR_INTEN | |
| W | | | | | | | | | | | | | | | | SWTRG |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DMAEN | PRESCALER | | | TRGSEL | | | | ENABLE | INTFLAG | INTEN | 0 | MULT | | CONT | LDOK |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PTIMER_STATUS_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-18**<br>**LDMODE** | RW | Load Mode Select<br>Selects the mode to load the MOD, IDLY, CHnDLYm, INTx, and PODLY registers, after 1 is written to LDOK.<br>00b - The internal registers are loaded with the values from their buffers, immediately after 1 is written to LDOK.<br>01b - The internal registers are loaded with the values from their buffers when the PTIMER counter (CNT) = MOD + 1 CNT delay elapsed, after 1 is written to LDOK.<br>10b - The internal registers are loaded with the values from their buffers when a trigger input event is detected, after 1 is written to LDOK.<br>11b - The internal registers are loaded with the values from their buffers when either the PTIMER counter (CNT) = MOD + 1 CNT delay elapsed, or a trigger input event is detected, after 1 is written to LDOK. |
| **17**<br>**SERR_INTEN** | RW | Sequence Error Interrupt Enable<br>Enables the PTIMER sequence error interrupt. When set, any of the PTIMER channel sequence error flags generates a PTIMER sequence error interrupt.<br>0b - PTIMER sequence error interrupt disabled.<br>1b - PTIMER sequence error interrupt enabled. |
| **16**<br>**SWTRG** | WO | Software Trigger<br>When PTIMER is enabled and the software trigger is selected as the trigger input source, writing 1 to SWTRG resets and restarts the counter. Writing 0 to SWTRG has no effect. Reading SWTRG yields 0. |

| | | |
|---|---|---|
| **15**<br>**DMAEN** | RW | DMA Enable<br>When DMA is enabled, the INTFLAG flag generates a DMA request instead of an interrupt.<br>0b - DMA disabled.<br>1b - DMA enabled. |
| **14-12**<br>**PRESCALER** | RW | Prescaler Divider Select<br>Counting uses the peripheral clock divided by the product of a factor (selected by MULT field) and an integer factor (set by PRESCALAR field), or in other words, (peripheral clock)/(MULT x PRESCALAR).<br>000b - Counting uses the peripheral clock divided by MULT (the multiplication factor).<br>001b - Counting uses the peripheral clock divided by 2 x MULT (the multiplication factor).<br>010b - Counting uses the peripheral clock divided by 4 x MULT (the multiplication factor).<br>011b - Counting uses the peripheral clock divided by 8 x MULT (the multiplication factor).<br>100b - Counting uses the peripheral clock divided by 16 x MULT (the multiplication factor).<br>101b - Counting uses the peripheral clock divided by 32 x MULT (the multiplication factor).<br>110b - Counting uses the peripheral clock divided by 64 x MULT (the multiplication factor).<br>111b - Counting uses the peripheral clock divided by 128 x MULT (the multiplication factor). |
| **11-8**<br>**TRGSEL** | RW | Trigger Input Source Select<br>Selects the trigger input source for the PTIMER. The trigger input source can be internal or external (EXTRG pin), or the software trigger. Refer to chip configuration details for the actual PTIMER input trigger connections.<br>0000b - Trigger-In 0 is selected.<br>0001b~1110b - Reserved<br>1111b - Software trigger is selected. |
| **7**<br>**ENABLE** | RW | Enable<br>0b - PTIMER disabled. Counter is off.<br>1b - PTIMER enabled. |
| **6**<br>**INTFLAG** | RW | PTIMER Interrupt Flag<br>INTFLAG is set when the counter value is equal to the IDLY register + 1.<br>• Write zero to clear INTFLAG.<br>• Writing 1 to INTFLAG has no effect. |

| | | |
|---|---|---|
| **5**<br>**INTEN** | RW | Interrupt Enable<br>Enables the PTIMER interrupt. When INTEN is set and DMAEN is cleared, INTFLAG generates a PTIMER interrupt.<br>0b - PTIMER interrupt disabled.<br>1b - PTIMER interrupt enabled. |
| **3-2**<br>**MULT** | RW | Multiplication Factor Select for Prescaler<br>Selects the multiplication factor of the prescaler divider for the counter clock.<br>00b - Multiplication factor is 1.<br>01b - Multiplication factor is 10.<br>10b - Multiplication factor is 20.<br>11b - Multiplication factor is 40. |
| **1**<br>**CONT** | RW | Continuous Mode Enable<br>Enables the PTIMER operation in Continuous mode.<br>0b - PTIMER operation in One-Shot mode<br>1b - PTIMER operation in Continuous mode |
| **0**<br>**LDOK** | RW | Load OK<br>Writing 1 to LDOK bit updates the MOD, IDLY, CHnDLYm, and POyDLY registers with the values previously written to their internal buffers (and stored there). The new values of MOD, IDLY, CHnDLYm, and POyDLY registers will take effect according to the setting of the LDMOD field (Load Mode Select).<br>Before 1 is written to the LDOK field, the values in the internal buffers of these registers are not effective, and new values cannot be written to the internal buffers until the existing values in the internal buffers are loaded into their corresponding registers.<br>• LDOK can be written only when PTIMEREN is set, or LDOK can be written at the same time when PTIMEREN is written to 1.<br>• LDOK is automatically cleared when the values in the internal buffers are loaded into the registers or when PTIMEREN bit (PTIMER Enable) is cleared.<br>• Writing 0 to LDOK has no effect. |

### 54.6.2 PTIMER_MAX_CNT (Max Count Number Register)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the STATUS_CTRL[LDOK] bit.

Offset : 4h

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

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MAX_CNT | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**PTIMER_MAX_CNT field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**MAX_CNT** | RW | Max Count Number<br>Specifies the period of the counter. When the counter reaches this value, it will be reset back to zero. If the PTIMER is in Continuous mode, the count begins anew. Reading this field returns the value of the internal register that is effective for the current cycle of PTIMER. |

### 54.6.3 PTIMER_CNT (Counter Register)

Note: Writing to this read-only register will not generate a transfer error or hard fault.

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CNT | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PTIMER_CNT field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**CNT** | RW | Counter Value<br>Contains the current value of the counter. |

### 54.6.4 PTIMER_INT_DLY (Interrupt Delay Register)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the STATUS_CTRL[LDOK] bit.

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INT_DLY | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**PTIMER_INT_DLY field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0 INT_DLY** | RW | |

### 54.6.5 PTIMER_CTRL_CHn (Channel n Control Register)

Each PTIMER channel has one control register, CHNnCTRL. The bits in this register control the functionality of each PTIMER channel operation.

Offset : 10h+N*28h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | BTB | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PTOS | | | | | | | | PTEN | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PTIMER_CTRL_CHn field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-16**<br>**BTB** | RW | Channel Pre-Trigger Back-to-Back Operation Enable<br>These bits enable the PTIMER ADC pre-trigger operation as back-to-back mode. Back-to-back operation enables the ADC conversions complete to trigger the next PTIMER channel pre-trigger and trigger output, so that the ADC conversions can be triggered on next set of configuration and results registers. Application code must only enable the back-to-back operation of the PTIMER pre-triggers at the leading of the back-to-back connection chain.<br>0b - PTIMER channel's corresponding pre-trigger back-to-back operation disabled.<br>1b - PTIMER channel's corresponding pre-trigger back-to-back operation enabled. |
| **15-8**<br>**PTOS** | RW | Channel Pre-Trigger Output Select<br>These bits select the PTIMER ADC pre-trigger outputs.<br>0b - PTIMER channel's corresponding pre-trigger is in bypassed mode. The pre-trigger asserts one peripheral clock cycle after a rising edge is detected on the selected trigger input source or software trigger is selected and SWTRG is written with 1.<br>1b - PTIMER channel's corresponding pre-trigger asserts when the counter reaches the channel delay register plus one peripheral clock cycle after a rising edge is detected on the selected trigger input source or software trigger is selected and SWTRG is written with 1. |
| **7-0**<br>**PTEN** | RW | Channel Pre-Trigger Enable<br>These bits enable the PTIMER ADC pre-trigger outputs.<br>0b - PTIMER channel's corresponding pre-trigger disabled.<br>1b - PTIMER channel's corresponding pre-trigger enabled. |

### 54.6.6 PTIMER_STATUS_CHn (Channel n Status Register)

Offset : 14h+N*28h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CHN_FLAG | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | SERR_FLAG | | | | | | | |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PTIMER_STATUS_CHn field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-16**<br>**CHN_FLAG** | RW | Channel Flags<br>The CF[m] field is set when the PTIMER counter (PTIMER_CNT) matches the value CHnDLYm + 1. Write 0 to clear CF. |
| **7-0**<br>**SERR_FLAG** | RW | PTIMER Channel Sequence Error Flags<br>0b - Sequence error not detected on PTIMER channel's corresponding pre-trigger.<br>1b - Sequence error detected on PTIMER channel's corresponding pre-trigger. ADCn block can be triggered for conversion by one pre-trigger from PTIMER channel n. When one conversion, which is triggered by one of the pre-triggers from PTIMER channel n, is in progress, a new trigger from PTIMER channel's corresponding pre-trigger m cannot be accepted by ADCn, and ERR[m] is set. Writing 0 to clear the sequence error flags. |

### 54.6.7 PTIMER_DLY0_CHn (Channel n Delay 0 Register)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the STATUS_CTRL[LDOK] bit.

Offset : 18h+N*28h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CHNDLY | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PTIMER_DLY0_CHn field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **15-0**<br>**CHNDLY** | RW | PTIMER Channel Delay<br>These bits specify the delay value for the channel's corresponding pre-trigger. The pre-trigger asserts when the counter is equal to DLY. Reading these bits returns the value of internal register that is effective for the current PTIMER cycle. |

### 54.6.8 PTIMER_DLY1_CHn (Channel n Delay 1 Register)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the STATUS_CTRL[LDOK] bit.

Offset : 1Ch+N*28h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CHNDLY1 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PTIMER_DLY1_CHn field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**CHNDLY1** | RW | PTIMER Channel Delay<br>These bits specify the delay value for the channel's corresponding pre-trigger. The pre-trigger asserts when the counter is equal to DLY. Reading these bits returns the value of internal register that is effective for the current PTIMER cycle. |

### 54.6.9 PTIMER_DLY2_CHn (Channel n Delay 2 Register)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the STATUS_CTRL[LDOK] bit.

Offset : 20h+N*28h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | CHNDLY2 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PTIMER_DLY2_CHn field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**CHNDLY2** | RW | PTIMER Channel Delay<br>These bits specify the delay value for the channel's corresponding pre-trigger. The pre-trigger asserts when the counter is equal to DLY. Reading these bits returns the value of internal register that is effective for the current PTIMER cycle. |

### 54.6.10 PTIMER_DLY3_CHn (Channel n Delay 3 Register)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the STATUS_CTRL[LDOK] bit.

Offset : 24h+N*28h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | CHNDLY3 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PTIMER_DLY3_CHn field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **15-0**<br>**CHNDLY3** | RW | PTIMER Channel Delay<br>These bits specify the delay value for the channel's corresponding pre-trigger. The pre-trigger asserts when the counter is equal to DLY. Reading these bits returns the value of internal register that is effective for the current PTIMER cycle. |

### 54.6.11 PTIMER_DLY4_CHn (Channel n Delay 4 Register)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the STATUS_CTRL[LDOK] bit.

Offset : 28h+N*28h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CHNDLY4 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PTIMER_DLY4_CHn field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**CHNDLY4** | RW | PTIMER Channel Delay<br>These bits specify the delay value for the channel's corresponding pre-trigger. The pre-trigger asserts when the counter is equal to DLY. Reading these bits returns the value of internal register that is effective for the current PTIMER cycle. |

### 54.6.12 PTIMER_DLY5_CHn (Channel n Delay 5 Register)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the STATUS_CTRL[LDOK] bit.

Offset : 2Ch+N*28h (N : 0 ~ 3)

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

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CHNDLY5 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PTIMER_DLY5_CHn field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**CHNDLY5** | RW | PTIMER Channel Delay<br>These bits specify the delay value for the channel's corresponding pre-trigger. The pre-trigger asserts when the counter is equal to DLY. Reading these bits returns the value of internal register that is effective for the current PTIMER cycle. |

### 54.6.13 PTIMER_DLY6_CHn (Channel n Delay 6 Register)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the STATUS_CTRL[LDOK] bit.

Offset : 30h+N*28h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CHNDLY6 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PTIMER_DLY6_CHn field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **15-0**<br>**CHNDLY6** | RW | PTIMER Channel Delay<br>These bits specify the delay value for the channel's corresponding pre-trigger. The pre-trigger asserts when the counter is equal to DLY. Reading these bits returns the value of internal register that is effective for the current PTIMER cycle. |

### 54.6.14 PTIMER_DLY7_CHn (Channel n Delay 7 Register)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the STATUS_CTRL[LDOK] bit.

Offset : 34h+N*28h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CHNDLY7 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PTIMER_DLY7_CHn field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**CHNDLY7** | RW | PTIMER Channel Delay<br>These bits specify the delay value for the channel's corresponding pre-trigger. The pre-trigger asserts when the counter is equal to DLY. Reading these bits returns the value of internal register that is effective for the current PTIMER cycle. |

### 54.6.15 PTIMER_POEN (Pulse-Out Enable Register)

Offset : 190h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | POEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PTIMER_POEN field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**POEN** | RW | PTIMER Pulse-Out Enable<br>Enables the pulse output. Only lower 1 bit are implemented in this MCU.<br>0b - PTIMER Pulse-Out disabled<br>1b - PTIMER Pulse-Out enabled |

### 54.6.16 PTIMER_PODLY (Pulse-Out Delay Register)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the STATUS_CTRL[LDOK] bit.

Offset : 194h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DLY1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DLY2 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**PTIMER_PODLY field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-16**<br>**DLY1** | RW | PTIMER Pulse-Out Delay 1<br>These bits specify the delay 1 value for the PTIMER Pulse-Out. Pulse-Out goes high when the PTIMER counter is equal to the DLY1. Reading these bits returns the value of internal register that is effective for the current PTIMER cycle. |

| | | |
|---|---|---|
| **15-0**<br>**DLY2** | RW | PTIMER Pulse-Out Delay 2<br>These bits specify the delay 2 value for the PTIMER Pulse-Out. Pulse-Out goes low when the PTIMER counter is equal to the DLY2. Reading these bits returns the value of internal register that is effective for the current PTIMER cycle. |

Flagchip

# Chapter 55 Frequency Measurement (FREQM)

## 55.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | Updated the description of the "MES_CLK_SEL" bit field in the register section *FREQM_CTRL* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 55.2 Chip-specific Information

N/A.

## 55.3 Features

- Support up to 64 input clock sources.
- Support pre-scaler.
- A reference counter counts on the reference clock.
- A measure counter counts on the measured clock.
- Support interrupt on each counter event.

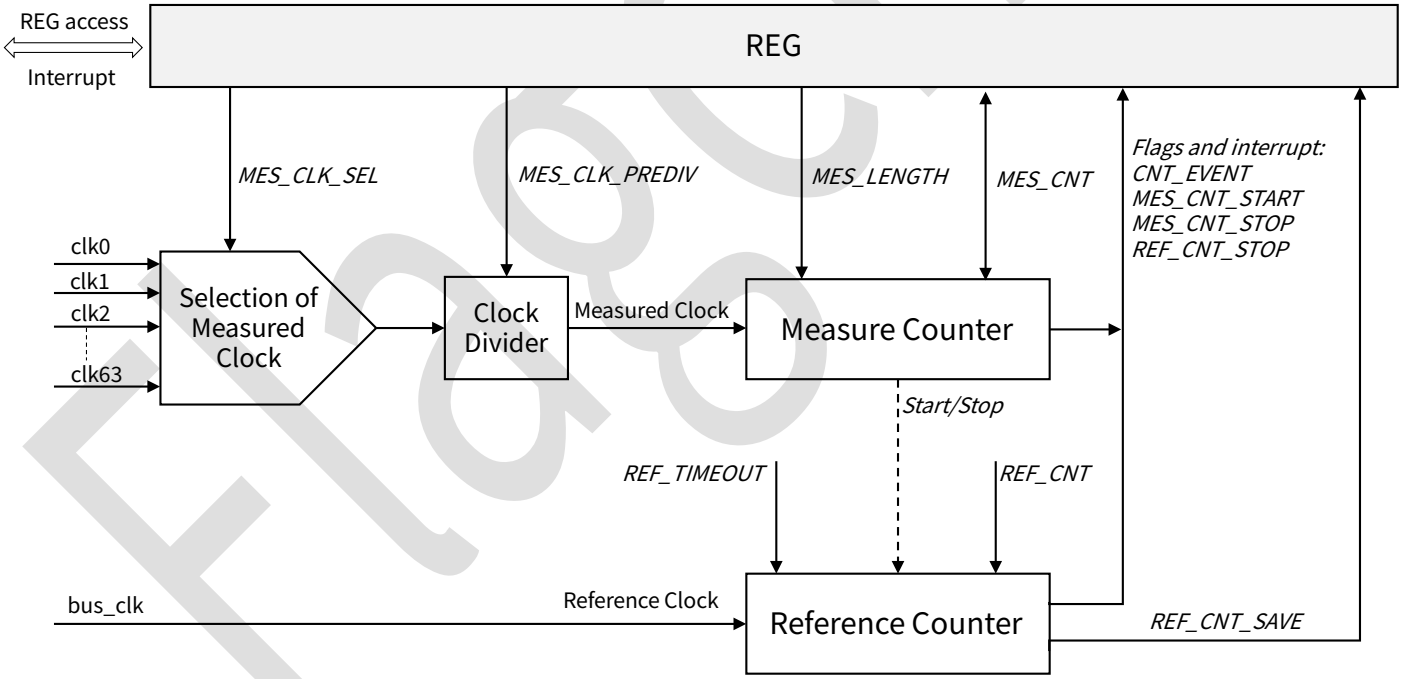
## 55.4 Overview

The Frequency Measurement (FREQM) module provides the capability to measure the frequency of on-chip clock sources. Up to 64 input clock sources can be selected and divided as the measured clock to drive the measure counter. On-chip bus_clk with a known frequency is used as the reference clock driving the reference counter. The frequency of the measured clock can be got by the counter values counted by the reference counter in a specified time window.

## 55.5 Block Diagram

The following is the block diagram of the FREQM module.

**Figure 122. FREQM block diagram**




## 55.6 Functional Description

### 55.6.1 Measured Clock Selection and Divider

The FREQM supports up to 64 input clock sources, and the clock to be measured is selected by *MES_CLK_SEL* in FREQM *CTRL* register. A clock divider is used to pre-scale the selected clock to a lower frequency, and the *MES_CLK_PREDIV* in FREQM *CTRL* register is used to set the divider value. To get higher accuracy, it is suggested that the frequency of the reference clock should be greater than the frequency of the measured clock.

### 55.6.2 Control Sequence

Follow the sequence below to do the frequency measurement:

1) Configure the clock divider value (*MES_CLK_PREDIV*).
2) Select the clock source to be measured (*MES_CLK_SEL*).
3) Configure the measure counter target value (*MES_LENGTH*). This will decide the time window for both measure counter and reference counter counting.
4) Configure the reference counter target value (*REF_TIMEOUT*). This will decide the timeout time. The timeout time must be long enough so that the measure counter can stop before the reference counter.
5) Clear reference/measure counter values and all statuses (*MES_CNT_START/ MES_CNT_STOP/REF_CNT_STOP*) by writing 0x00000000 to FREQM *REF_CNT* register. And wait for all statuses to be 0.
6) Start reference/measure counter by writing 0x00000000 to FREQM *MES_CNT* register. The counter value of the reference counter will be saved in *REF_CNT_SAVE* when the measure counter starts and stops.
7) Check FREQM *CNT_STATUS* register by polling or interrupt if *CNT_EVENT_IE* is set. Once *CNT_STATUS[2:0]* is 0x4 which means the measure counter has started, read *REF_CNT_SAVE* and save it as N_START.
8) Check FREQM *CNT_STATUS* register by polling or interrupt if *CNT_EVENT_IE* is set. Once *CNT_STATUS[2:0]* is 0x6 which means the measure counter has stopped, read *REF_CNT_SAVE* and save it as N_STOP. The frequency of the measured clock can be calculated by:
   DIV* F_REF * (MES_LENGTH-1) / (N_STOP- N_START) where F_REF is reference clock frequency and DIV is decided by MES_CLK_PREDIV.
9) Check FREQM *CNT_STATUS* register by polling or interrupt if *CNT_EVENT_IE* is set. Once *CNT_STATUS[2:0]* is 0x7, both reference/measure counters have stopped. The next measurement can be continued. It supposes that the measure counter stop before the reference counter stops, so if *CNT_STATUS[0]* is 1, but *CNT_STATUS[2:1]* is not 3, which means the measure counter has not stopped when the reference counter stops. In this situation, we should check if the timeout time (*REF_TIMEOUT*) is long enough or if the measured clock is off.

### 55.6.3 Interrupt

The FREQM supports operation by interrupts, the enable is controlled by *CNT_EVENT_IE* bit in FREQM *CTRL* register. An interrupt flag *CNT_EVENT* is in FREQM *CNT_STATUS* register. An interrupt can be generated:

- When the measure counter starts (MES_CNT_START assert).
- When the measure counter stops (MES_CNT_STOP assert).
- When the reference counter starts (REF_CNT_START assert).

The interrupt of FREQM is a synchronous interrupt, it can only be used in run and wait mode.

### 55.6.4 Reset

The FREQM has two external reset sources: Power-on Reset (POR) and system reset. Register FREQM *CTRL* can only be reset by POR. Other registers and all function clock domain logics can be reset by the two reset sources. The FREQM has one internal synchronous reset source, triggered automatically when FREQM *CTRL* is written, used to reset control logic.

### 55.6.5 Stop Mode

When FREQM is in the process of measuring a clock and there comes a stop mode entry request, the current FREQM will send the stop mode acknowledge signal only after the measure counter stops (*CNT_STATUS[2:0] = 0x6)*. The N_STOP can be read from *REF_CNT_SAVE* for frequency calculation after stop mode exit.

## 55.7 Register Definition

FREQM base address : 4007_8000h

**Register Overview - FREQM**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | FREQM_CTRL (Control Register) | 32 | RW | 0000_0000h | page |
| **4h** | FREQM_MES_CNT (Measure Counter Register) | 32 | RW | 0000_0000h | page |
| **8h** | FREQM_REF_CNT (Reference Counter Register) | 32 | WO | 0000_0000h | page |
| **Ch** | FREQM_MES_LENGTH (Measure Counter Length Register) | 32 | RW | FFFF_FFFFh | page |
| **10h** | FREQM_REF_TIMEOUT (Reference Counter Timeout Register) | 32 | RW | FFFF_FFFFh | page |
| **14h** | FREQM_CNT_STATUS (Counter Status Register) | 32 | RW | 0000_0000h | page |
| **1Ch** | FREQM_REF_CNT_SAVE (Saved Reference Counter Register) | 32 | RO | 0000_0000h | page |

### 55.7.1 FREQM_CTRL (Control Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | CNT_EVENT_IE | MES_CLK_SEL | | | | | | MES_CLK_PREDIV | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FREQM_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **14**<br>**CNT_EVENT_IE** | RW | Counter Event Interrupt Enable<br>0b - Interrupt is disabled<br>1b - Interrupt is enabled |

| | | |
|---|---|---|
| **13-8**<br>**MES_CLK_SEL** | RW | Measured Clock Selection<br>0: Reserved<br>1: Slow Clock<br>2: PLL1_FEEDBACK<br>3: PLL0_FEEDBACK<br>4: SCG_CLKOUT<br>5: RTC_CLK<br>6: AON_CLK<br>7: SIRC128K<br>8: Reserved<br>9: Reserved<br>10: FCPIT0 functional clock<br>11: Reserved<br>12: Reserved<br>13: FCSPI0 functional clock<br>14: FCSPI1 functional clock<br>15: FCSPI2 functional clock<br>16: Reserved<br>17: FCUART0 functional clock<br>18: FCUART1 functional clock<br>19: FCUART2 functional clock<br>20: Reserved<br>21: TRGSEL1_TRGSEL_CLK<br>22: FTU0 functional clock<br>23: FTU1 functional clock<br>24: Reserved<br>25: FTU2 functional clock<br>26: FTU3 functional clock<br>27: ADC0 functional clock<br>28: Reserved<br>29: ADC1 functional clock<br>30: Reserved<br>31: Reserved<br>32-39: DIVL[0:7]<br>40:47: DIVM[0:7]<br>48-55: DIVH[0:7]<br>56: Reserved<br>57: SENT0 functional clock<br>58: FLEXCAN0 functional clock<br>59: FLEXCAN1 functional clock<br>60: Reserved<br>61: MSC0 functional clock<br>62: TPU functional clock<br>63: Reserved |

Flagchip

| | | |
|---|---|---|
| **7-0**<br>**MES_CLK_PREDIV** | RW | Measure clock pre_div. Divider value is (MES_CLK_PREDIV>>1)*2. |

### 55.7.2 FREQM_MES_CNT (Measure Counter Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MES_CNT | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MES_CNT | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FREQM_MES_CNT field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**MES_CNT** | RW | Value of Measure Counter<br>Will read all 0 if measure counter has not stopped.<br>Write 0 to start both reference and measure counters. |

### 55.7.3 FREQM_REF_CNT (Reference Counter Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | REF_CNT | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | REF_CNT | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FREQM_REF_CNT field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 REF_CNT** | WO | Value of Reference Counter<br>Writes 0 to clear reference and measure counter values and all status flags (MES_CNT_START/ MES_CNT_STOP/REF_CNT_STOP). |

### 55.7.4 FREQM_MES_LENGTH (Measure Counter Length Register)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MES_LENGTH | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MES_LENGTH | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**FREQM_MES_LENGTH field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 MES_LENGTH** | RW | Counting Length of Measure Counter<br>This will decide the time window for both measure counter and reference counter counting. |

### 55.7.5 FREQM_REF_TIMEOUT (Reference Counter Timeout Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | REF_TIMEOUT | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | REF_TIMEOUT | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**FREQM_REF_TIMEOUT field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**REF_TIMEOUT** | RW | Timeout Value of Reference Counter<br>This will decide the timeout time. The timeout time must be long enough so that the measure counter can stop before reference counter. |

### 55.7.6 FREQM_CNT_STATUS (Counter Status Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | CNT_EVENT | 0 | | | | MES_CNT_START | MES_CNT_STOP | REF_CNT_STOP |
| W | | | | | | | | | w1c | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FREQM_CNT_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **7**<br>**CNT_EVENT** | W1C | Counter Event Interrupt Flag<br>Assert when MES_CNT_START/MES_CNT_STOP/REF_CNT_STOP assert.<br>Write 1 to clear (W1C) |
| **2**<br>**MES_CNT_START** | RO | Measure Counter Start |
| **1**<br>**MES_CNT_STOP** | RO | Measure Counter Stop |
| **0**<br>**REF_CNT_STOP** | RO | Reference Counter Stop |

### 55.7.7 FREQM_REF_CNT_SAVE (Saved Reference Counter Register)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REF_CNT_SAVE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | REF_CNT_SAVE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FREQM_REF_CNT_SAVE field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**REF_CNT_SAVE** | RO | Saved Reference Counter Value<br>Save at the measure counter starts and stops. |

Flagchip

# Chapter 56 Time Processing Unit (TPU)

## 56.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | Updated the description of the "ERS" bit field in the register section *TPU_E_CHn_ER1* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 56.2 Chip-specific Information

### 56.2.1 Chip Mode and TPU Register Side

FC7240F2M contains a Time Processing Unit (TPU) module. This module has two AFCB slots, divided into the host side and the engine side. These two sides have different access rights in different modes of the chip. See the table below for details:

| FlexCore Mode | **Host CPU** | **Engine CPU** |
|---|---|---|
| **Engine side** | No | Yes |
| **Host side** | Yes | No |

| Non-FlexCore Mode | **Host/Engine CPU** |
|---|---|
| **Engine side** | Yes |
| **Host side** | Yes |

***Note:*** In non-FlexCore mode, the TPU's Host CPU and Engine CPU are the same.

### 56.2.2 TPU Clocking

TPU has two clocks, including bus clock and functional clock. The opening method of function clock varies in different modes.

- FlexCore Mode: Host CPU needs to turn on the Engine CPU subsystem clock first, and then Engine CPU turn on the TPU clock.
- Non-FlexCore Mode: Host CPU needs to turn on the TPU clock.

### 56.2.3 TPU Reset

TPU has a reset signal, it is different in different modes.

- FlexCore Mode: The reset of the TPU is the same as the reset of the Engine CPU subsystem, and Engine CPU can soft reset the TPU through the configuration register.
  Host CPU can soft reset the Engine CPU subsystem through SCM_SUBSYS_PCC[SWRST].
- Non-FlexCore Mode: The reset of the TPU is the same as the reset of the Host CPU subsystem.

### 56.2.4 TPU Debug

- FlexCore Mode: Only when Engine CPU is in Debug mode, TPU is in Debug mode.
- Non-FlexCore Mode: Only when Host CPU is in Debug mode, TPU is in Debug mode.

### 56.2.5 TPU Global Time Base (GTB)

The TPU supports a global time base through the Global Time Base (GTB) feature. The global time base allows the TPU counters to start their operation synchronously. Global Time Base Input (GTBIN) of the TPU is connected to SCM_FTU_GTBC[TPU_GTBC].

In addition, the four channel trigger signals of TSTMP1 can also enable TPU's global time base, and users can configure SCM_TPU_GTBCM[TPU_GTBCM] determines which channel of TSTMP1 triggers the signal to enable the global time base of the TPU, causing the TPU counter to start to work.

### 56.2.6 TPU Channel Input

The channel inputs are connected to:

- External pins.
- TRGSEL5 output.

Flagchip

The TPU has a total of 32 channels, of which the input signals of the last 8 channels 24-31 are optional. Users can configure SCM_TPU_ GTBCM [TPU_TRGSEL] to select whether the channel input source is pad or trgsel5 output.

The TCRCLK input is connected to the same input as channel 0.

### 56.2.7 TPU Channel Output

The channel outputs are connected to:

- External pins.
- TRGSEL input.

TPU has four FAULTIN input signals, which are derived from the output of TRGSEL2 and are used to control the output of the channel. FAULT0 controls the 8 channels of the TPU, which are 0-7 respectively. FAULT1/2/3 are similar, covering a total of 32 channels.

## 56.3 Features

- Two 24-bit time counters.
- Support 32 channels.
- Support input capture and match event.
- Channels used to trigger the ADC Subsystem.
- Channels with DMA request.
- Channels with Interrupt request and priority configurable.
- Channel output buffers are controllable.
- Support predefined channel mode and User Defined channel mode.
- Support fast interrupt response through entry table/entry points in FlexCore mode.
- Support HSR/HSA.
- Support Angle count mode.
- Support software reset.

## 56.4 Overview

TPU is an intelligent, semi-autonomous module designed for timing control. TPU requires the engine CPU (FlexCore mode) to work together to achieve parallel work with the Host CPU. If only some simple scenarios are applied, the Host CPU can also be used as the engine CPU (Non-FlexCore) to achieve time processing control.

In FlexCore mode, Engine CPU processes instructions, accesses data memory and configures the TPU. TPU processes input events in real time and performs output waveform generation. Consequently, for each timer event, the Host CPU setup and service times are minimized or eliminated. At this time, Engi serves as the control core of the TPU and is open to users, allowing users to develop their own functions on the TPU.

In Non-FlexCore mode, the Host/Engine CPU processes instructions, accesses data memory and configures the TPU. TPU processes input events in real time and performs output waveform generation. Since the Host CPU also participates in other task scheduling, the HOST CPU cannot completely serve as the control core of the TPU. The use of the TPU in this mode needs to consider user application requirements.

TPU architecture aims at high resolution timing capabilities, high resolution timing is achieved by two main capabilities:

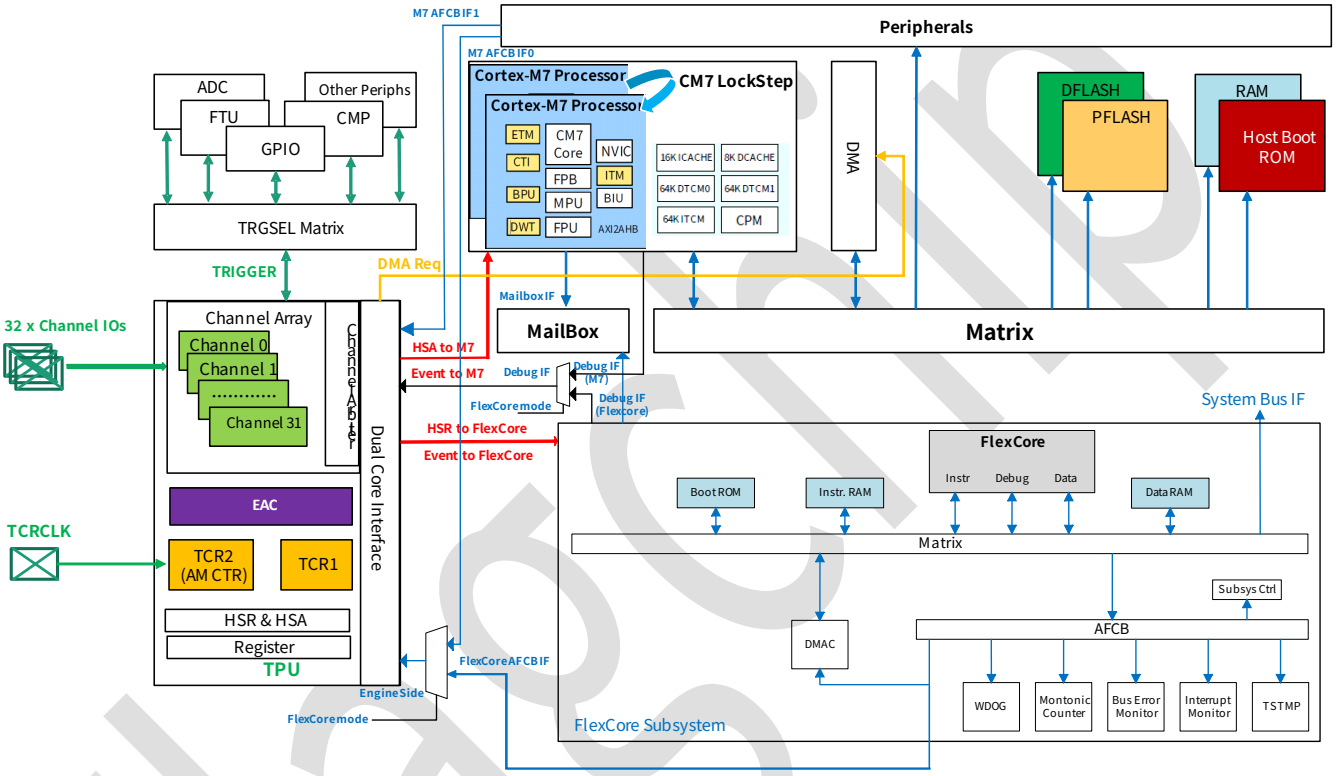
- Reduced latency: pin actions are immediate.
- Double action channel capability reducing the channel request rate.

In FlexCore mode, TPU and Engine CPU provides higher resolution than the Host CPU can achieve and creates no Host overhead for servicing timing tasks. Latency is the interval from occurrence of an event to the start of event servicing. Engine CPU can service TPU events without interrupting the Host. There are two types of timing events:

- Input pin transition.
- Selected Time Base match, ie., a selected Time Base counter reached or exceeded a pre-programed value.

The figure below shows a top-level TPU system diagram, which displays the connection of TPU in Soc system.

**Figure 123. TPU system diagram**



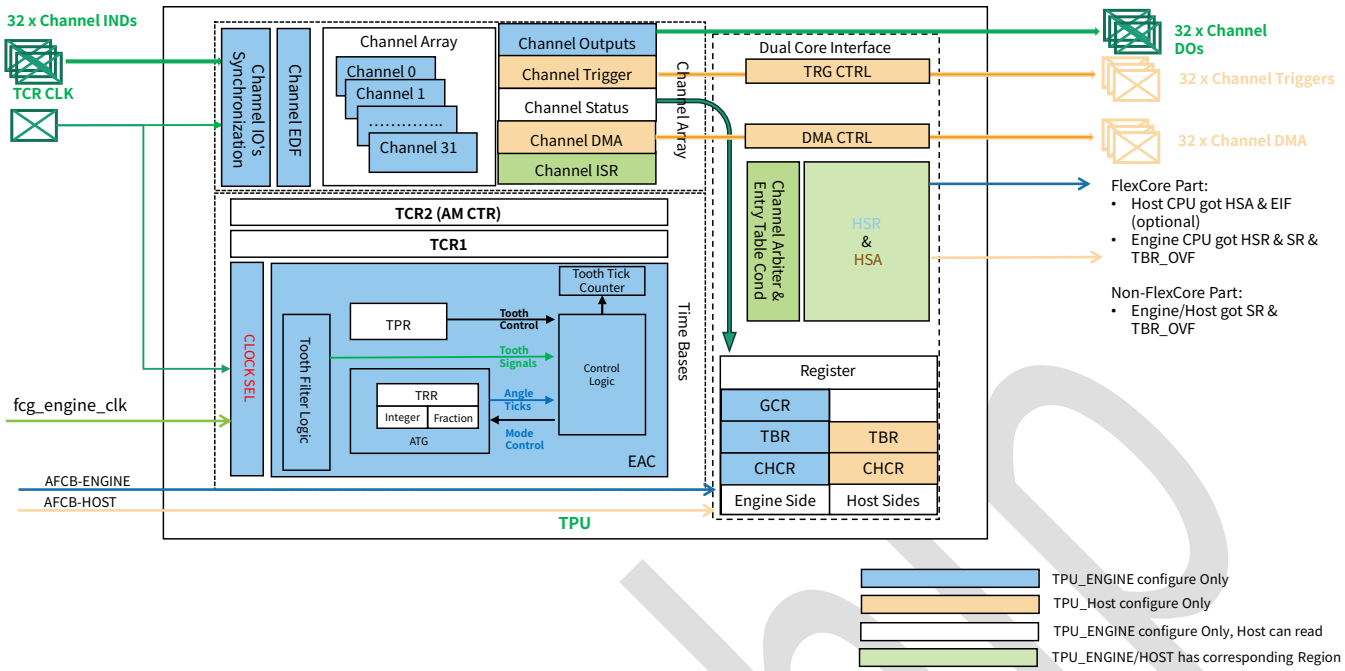

TPU is responsible for processing input pin transitions and output pin waveform generation based on the **Time Bases**.

**FlexCore** mode includes the FlexCore subsystem, which is used as the control core of the TPU and works in parallel with the Host CPU.

**Non-FlexCore** mode, the Host CPU controls the TPU work. At this time, the Engine CPU and Host CPU of the TPU are the same.

The following figure shows the block diagram for the TPU.

**Figure 124. TPU block diagram**




**Time Bases**:

Two 24-bit counters TCR1 and TCR2 provide reference time bases for all match and input capture events. Prescalers for both time bases are controlled by the Engine CPU through bit fields in the TPU configuration registers. The TPU is able to export/import time to/from TCR1 or TCR2. The clock for each of TCR1 and TCR2 clock can be independently derived from the TPU clock or from an external input via the TCRCLK clock pin. In addition, the TCR2 timebase can be derived from special angle-clock hardware which enables implementing angle-based functions. This feature is added to support advanced angle based engine control applications. For further details refer to Time Bases.

**TPU Timer Channel**:

Each TPU Engine has 32 independent channels, each corresponding to an Input/Output signal pair. The channel time resolution is 24 bits, and are all identical. Each channel consists of logic which supports two events and output controls. The event logic contains two 24-bit capture registers, two 24-bit match registers, greater-equal and equal-only comparators. Supporting two events enables many combinations of double action functions (for example the channel can handle two events with a single microcode service). The channel configuration can be changed immediately by the Engine CPU. Each channel can perform double capture, double match and other capture-match combinations. Channel modes available can do ordered or unordered match. Some modes are also provided that can block one match by the occurrence of the other. Service request can be generated on one or both of the match events. Input signal can be separated from output signal in each channel. They can, optionally, be combined in a single I/O pin driver. An output buffer enable is controlled by Engine CPU. Digital filters are provided for the input signals, with distinct filtering modes available. Each channel can use any time base or angle counter for either match or capture operation. For example, a match on TCR1 can capture the value of TCR2. The channels can request service from the Engine CPU due to recognized pin transitions (input events) or time-base matches.

Channel configuration combinations:

- Single input capture, no match
- Single input capture with single match timeout
- Single input capture with double match timeout with several double match sub-modes

- Double input capture with single or double match timeout with several double match sub-modes
- Single output match
- Double output match with several double match sub-modes
- Input-dependent output generation

The double match functionality has various combinations for generation of service request and determining pin actions. For more details refer to Enhanced Channels. In addition to the predefined channel configurations above, the user can also program its own channel configuration, defining how input captures, matches and service-requests are related.

**EAC:**

The TPU contains an enhance angle counter hardware, has provisions for fast tick counting plus compensation for missing teeth, which are often used to provide an angle key for the shaft. The hardware also provides automatic compensation for acceleration and deceleration, and assists in error recovery. The software handles synchronization and exception handling.

**Host Service:**

TPU provides host service, used for fast communication between Host CPU and Engine CPU. Note that host service is only valid in FlexCore mode.

## 56.5 Functional Description

### 56.5.1 System Mode and Resource

In Non-FlexCore mode, the TPU's Engine CPU and Host CPU are the same.

**FlexCore Mode:**

REGISTER:

Engine CPU accesses engine side registers, Host CPU accesses host side registers.

INTERRUPT:

1) Channel interrupts are connected to Engine CPU, support interrupt priority configuration, support fast response through Entry table/function, The user can disable interrupts fed into Engine CPU via the configuration register.
2) Channel interrupts are connected to Host CPU at the same time. Channels 0-7 are one interrupt number and do not support fast response, The user can disable interrupts fed into Host CPU via the configuration register.
3) Host service request interrupt connects to Engine CPU. This interrupt indicates that Host CPU sends a request to Engine CPU.
4) Host service acknowledge interrupt is connected to Host CPU. This interrupt indicates that Engine CPU responds to Host CPU.

DMA_REQ:

Channel DMA request is connected to DMA0 of the Host CPU subsystem.

**Non-FlexCore Mode:**

REGISTER:

Host CPU accesses engine side and host side registers.

INTERRUPT:

1) Channel interrupts are connected to Host CPU at the same time. Channels 0-7 are one interrupt number and do not support fast response, The user can disable interrupts fed into Host CPU via the configuration register.
2) TCR1/2 counter overflow interrupt to Host CPU.

DMA_REQ:

Channel DMA request is connected to DMA0 of the Host CPU subsystem.

### 56.5.2 Function and ISR

TPU processing is event-driven, in the sense that TPU Engine CPU only runs to service a request from an event. Service Requests may result from the occurrence of any of the following events:

- Type1: Host CPU sent a HSR (Host Service Request) by configuring CHn_CR[CHSR] and CHn_SCR[CHRT].
- Type2: Occurrence of a time base match, an input signal transition, or a specific combination of them (depending on the Channel Mode currently configured).

A given event is always associated to only one Channel:

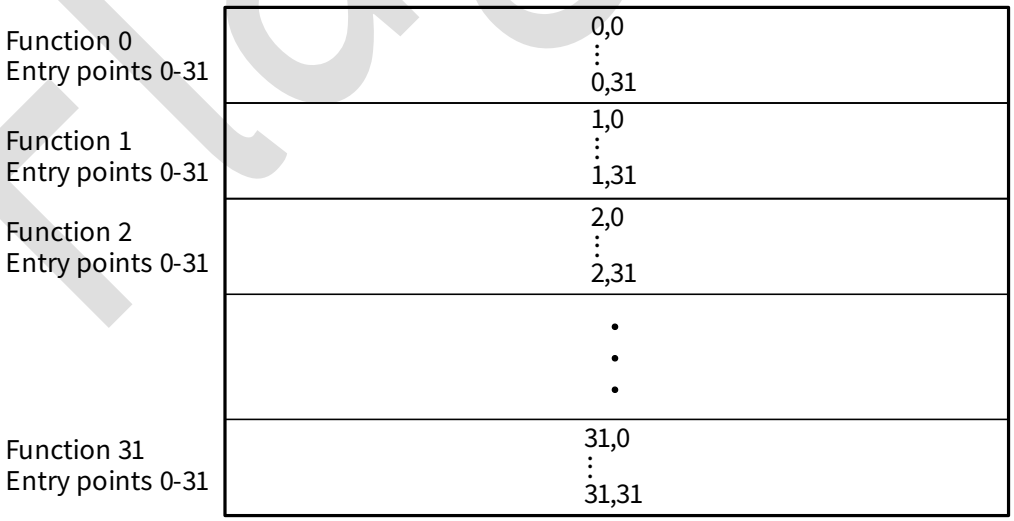
- Type1: There is CR[CHSR] and SCR[CHRT] Register field for each Channel
- Type2: Each signal is associated with only one Channel, which has its own Match registers and independent mode configuration.

Service Request processing is done by a set of Engine CPU routines. A set of related routines that implement a specific channel application is called a Function. One or more Functions reside on Data Memory in FlexCore Subsystem, limited only by the FlexCore DRAM space available, size of Engine CPU Functions and the number of entry points available.

A Function can be assigned to several channels, but only one Function can be assigned to a given Channel at a time. This is defined by the Engine CPU through the Channel Configuration Registers.

TPU supports 32 functions, and each function supports 32 entry points, thus forming an entry table. The storage location of the Entry table is determined by the user.

**Figure 125. Storage location of entry table**




A given Channel Function is selected and called by the TPU channel interrupt arbiter to decide. Mainly depends on the following information:

- Global arbiter priority rules (user configurable)

- Channel function selection and priority configuration (user configurable)

When channels interrupt comes, the interrupt arbiter sorts according to the priority of the functions configured for each channel and sends the interrupt to the engine CPU. Engine CPU enters the interrupt ISR and executes the corresponding function based on the results of the arbiter (channel ID, selected function, and conditions). Execute the corresponding entry point in the function according to the obtained conditions.

#### 56.5.2.1 Entry point

Each Function has its Entry point, up to 32 entry points. The entry point mainly consists of conditions, mainly including the following aspects:

- Channel Flags0 and 1
- Input pin state or Output flip flop
- Match1/Trans2 events
- Match2/Trans1 events
- Host Service Request events
- Entry table condition encoding scheme selection

Channel Flags 0 and 1: these are channel-internal flags associated with a channel. Their values are set by user (see Channel Registers).

Input Pin state or Output Flip Flop: the state (0 or 1) of the channel input signal after the Enhanced Filter (see Enhanced Digital Filter - EDF), or the state driven to the output signal. Which one (input or output) is used is selected by the CHn_CR[ETPD].

The Match1/Trans2 refer to the recognition of either a Match event specified by CHn_ER1 register or the detection of a channel input signal event specified by the CHn_CR2[IPAC2].

The Match2/Trans1 refer to the recognition of either a Match event specified by CHn_ER2 register or the detection of a channel input signal event specified by the CHn_CR2[IPAC1].

The Host Service Request refer to the value written by the Host CPU to the CHn_CR[CHSR] and CHn_SCR[CHRT] Register of the Channel being serviced. If the value of HSR is not zero, then the Host actually requested service.

The two Entry Table Condition encoding schemes combine events and state conditions differently, as detailed in the following sections.

##### 56.5.2.1.1 Standard Condition Encoding Scheme

In this scheme, shown in the following table, all 3 HSR combinations are used and other event type columns are marked "x" when HSR is non-zero, indicating that Host Service Request has priority over any other type of event. However, when an HSR service thread is called (entry point numbers 0 to 9), other events may also have been recognized, and it is Engine CPU responsibility to check them. When HSR is 0, i.e., Host did not issue a Service Request to the channel, the other event conditions, the input signal state and channel flags determine the Entry Point.

**Table 75. Standard channel condition encoding scheme**

| No. | Encoded Channel Conditions | Host Service Request Bits | Match1/ Trans2 | Match2/ Trans1 | In/Output Pin State | Channel Flag1 | Channel Flag0 |
|---|---|---|---|---|---|---|---|
| 0 | 00000 | 001 | x | x | 0 | - | 0 |
| 1 | 00001 | 001 | x | x | 0 | - | 1 |

| No. | Encoded Channel Conditions | Host Service Request Bits | Match1/ Trans2 | Match2/ Trans1 | In/Output Pin State | Channel Flag1 | Channel Flag0 |
|---|---|---|---|---|---|---|---|
| 2 | 00010 | 001 | x | x | 1 | x | 0 |
| 3 | 00011 | 001 | x | x | 1 | x | 1 |
| 4 | 00100 | 010 | x | x | x | x | x |
| 5 | 00101 | 011 | x | x | x | x | x |
| 6 | 00110 | 100 | x | x | x | x | x |
| 7 | 00111 | 101 | x | x | x | x | x |
| 8 | 01000 | 110 | x | x | x | x | x |
| 9 | 01001 | 111 | x | x | x | x | x |
| 10 | 01010 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| 11 | 01011 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| 12 | 01100 | 000 | 0 | 1 | 0 | x | 0 |
| 13 | 01101 | 000 | 0 | 1 | 0 | x | 1 |
| 14 | 01110 | 000 | 0 | 1 | 1 | x | 0 |
| 15 | 01111 | 000 | 0 | 1 | 1 | x | 1 |
| 16 | 10000 | 000 | 1 | 0 | 0 | x | 0 |
| 17 | 10001 | 000 | 1 | 0 | 0 | x | 1 |
| 18 | 10010 | 000 | 1 | 0 | 1 | x | 0 |
| 19 | 10011 | 000 | 1 | 0 | 1 | x | 1 |
| 20 | 10100 | 000 | 1 | 1 | 0 | x | 0 |
| 21 | 10101 | 000 | 1 | 1 | 0 | x | 1 |
| 22 | 10110 | 000 | 1 | 1 | 1 | x | 0 |
| 23 | 10111 | 000 | 1 | 1 | 1 | x | 1 |
| 24 | 11000 | 000 | 0 | 0 | 0 | x | 0 |
| 25 | 11001 | 000 | 0 | 0 | 0 | x | 1 |
| 26 | 11010 | 000 | 0 | 0 | 1 | x | 0 |
| 27 | 11011 | 000 | 0 | 0 | 1 | x | 1 |
| 28 | 11100 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| 29 | 11101 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| 30 | 11110 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |

Flagchip

| No. | Encoded Channel Conditions | Host Service Request Bits | Match1/ Trans2 | Match2/ Trans1 | In/Output Pin State | Channel Flag1 | Channel Flag0 |
|---|---|---|---|---|---|---|---|
| 31 | 11111 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |

#### 56.5.2.1.2 Alternate Condition Encoding Scheme

This scheme is shown in the following table. Because the HSR bits cannot be tested by Engine CPU, only three distinct Host Service Request can be used:

1) HSR=010 or 011, which are coded into the same Entry Points (0 to 3)
2) HSR=100,101 or 001, which are all coded into Entry Point 4
3) HSR=110 or 111, which are both coded into Entry Point 5

The remaining Entry Points use both channel flags for better state decoding, making this scheme better suited for Functions which need more states and/or faster state decoding, without needing many HSRs.

**Table 76. Alternate channel condition encoding scheme**

| No. | Encoded Channel Conditions [C4-C0] | Host Service Request Bits | Match1/ Trans2 | Match2/ Trans1 | In/Output Pin State | Channel Flag1 | Channel Flag0 |
|---|---|---|---|---|---|---|---|
| 0 | 00000 | 01x | x | x | 0 | x | 0 |
| 1 | 00001 | 01x | x | x | 0 | x | 1 |
| 2 | 00010 | 01x | x | x | 1 | x | 0 |
| 3 | 00011 | 01x | x | x | 1 | x | 1 |
| 4 | 00100 | 10x/001 | x | x | x | x | x |
| 5 | 00101 | 11x | x | x | x | x | x |
| 6 | 00110 | 000 | 0 | 0 | 0 | x | x |
| 7 | 00111 | 000 | 0 | 0 | 1 | x | x |
| 8 | 01000 | 000 | 1 | 0 | 0 | 0 | 0 |
| 9 | 01001 | 000 | 1 | 0 | 0 | 0 | 1 |
| 10 | 01010 | 000 | 1 | 0 | 0 | 1 | 0 |
| 11 | 01011 | 000 | 1 | 0 | 0 | 1 | 1 |
| 12 | 01100 | 000 | 1 | 0 | 1 | 0 | 0 |
| 13 | 01101 | 000 | 1 | 0 | 1 | 0 | 1 |
| 14 | 01110 | 000 | 1 | 0 | 1 | 1 | 0 |
| 15 | 01111 | 000 | 1 | 0 | 1 | 1 | 1 |
| 16 | 10000 | 000 | 0 | 1 | 0 | 0 | 0 |
| 17 | 10001 | 000 | 0 | 1 | 0 | 0 | 1 |

| No. | Encoded Channel Conditions [C4-C0] | Host Service Request Bits | Match1/ Trans2 | Match2/ Trans1 | In/Output Pin State | Channel Flag1 | Channel Flag0 |
|---|---|---|---|---|---|---|---|
| 18 | 10010 | 000 | 0 | 1 | 0 | 1 | 0 |
| 19 | 10011 | 000 | 0 | 1 | 0 | 1 | 1 |
| 20 | 10100 | 000 | 0 | 1 | 1 | 0 | 0 |
| 21 | 10101 | 000 | 0 | 1 | 1 | 0 | 1 |
| 22 | 10110 | 000 | 0 | 1 | 1 | 1 | 0 |
| 23 | 10111 | 000 | 0 | 1 | 1 | 1 | 1 |
| 24 | 11000 | 000 | 1 | 1 | 0 | 0 | 0 |
| 25 | 11001 | 000 | 1 | 1 | 0 | 0 | 1 |
| 26 | 11010 | 000 | 1 | 1 | 0 | 1 | 0 |
| 27 | 11011 | 000 | 1 | 1 | 0 | 1 | 1 |
| 28 | 11100 | 000 | 1 | 1 | 1 | 0 | 0 |
| 29 | 11101 | 000 | 1 | 1 | 1 | 0 | 1 |
| 30 | 11110 | 000 | 1 | 1 | 1 | 1 | 0 |
| 31 | 11111 | 000 | 1 | 1 | 1 | 1 | 1 |

***Note:*** This part is only valid in FlexCore mode.

### 56.5.3 Enhanced Channels

Enhanced Channels comprise hardware support for input digital signal processing and output signal generation. Each Channel is associated with one input and one output signal. Enhanced Channel logic is combined with Function Engine CPU (and optionally Angle Mode logic) to implement Channel I/O functionality.

TPU Enhanced Channels are capable of dual action, meaning that each channel logic can handle two events at different times and/or cause two separated actions – these actions and events can be mutually dependent (with the first either blocking or enabling the other), or both independent, depending on the programmed Channel Mode.

Each Enhanced Channel contains event logic containing two Event Register sets, each set supporting one input and/or output action, the pair implementing dual action support. Each Event Register set contains two 24-bit registers: Match and Capture. The Match register holds the pending match value which is compared against one of the two time bases by an equal-only/greater-equal comparator. The Capture register captures one of the two time bases as a result of a Match or Transition detection. Service Requests are issued on particular combination of match and capture events, defined by the selected Channel Mode.

In the context of the TPU channels, a Match is a comparison between a time base value and a channel Match register. If those two values are coincident, or the time base value is greater than the value of the Match register, a Match Event occurs. Depending on the channel mode of operation and current state of the channel logic, the match event may be recognized, i.e., change the state of the channel, or be ignored. A match event recognized by the channel logic is called a Match Recognition. Match Recognitions can cause, also depending on Channel Mode and current state, the channel to request service, configuring a Match Service Request.

TPU uses two kinds of comparator to assert a Match Event: an Equal comparator, in which both the Match Register and the value of the selected time base must match exactly, and a Greater-Equal comparator. The Greater-Equal comparator considers any time base value between the range [N : N+0x800000-1] as a valid match against the value of N in the Match Register, even when the value N+0x800000-1 wraps around the point of origin (0x0). Refer to the following figure for an illustration of the matching values on a Greater-Equal comparator.

The second source of events for the TPU channel is a Transition detected at the corresponding channel's input signal. Two distinct Transition detections can be programmed individually for each channel, allowing recognition of all possible combinations of edge detection. It is also possible to check the sampled state of an input signal upon the occurrence of a Match: the sampling of the expected value is treated as a Transition, even if the input signal did not necessarily toggled at the time of the Match, or at any time at all.

Like Match Events, Transitions Events may or not be recognized by the channel logic. When they are, a Transition Detection occurs. As well as Match Recognitions, Transition Detections can issue a Channel Service Request, depending on Channel Mode and current state.

Transition Detections and Match Recognitions are sometimes simply called Transitions and Matches throughout this document, for short.

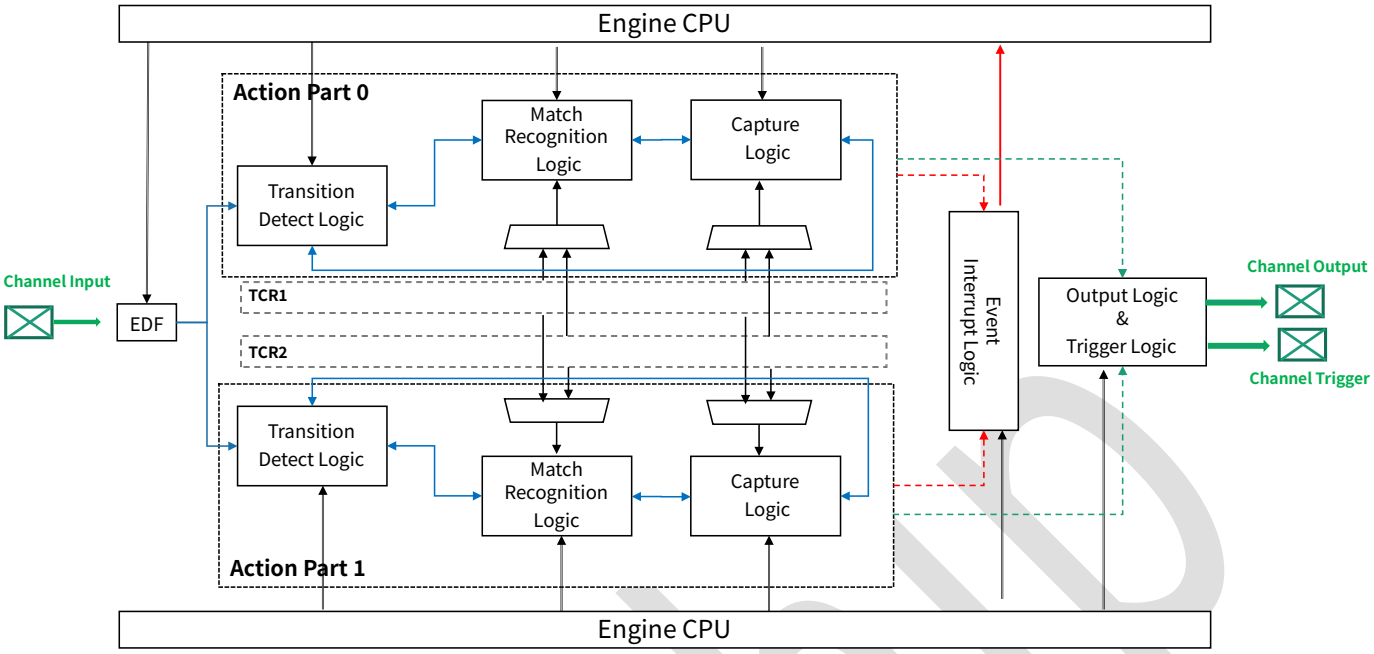
Input and output signals can be processed separately by the channel logic and user-code and can also be combined such that Matches and Transitions are used to cause output signal actions. The output signals can also be directly controlled by user-code. Many event combinations are allowed for a channel, given the possibility of configuring pairs of matches and transitions for the dual action logic, where each event is able to block or enable the next one. There is a full set of Channel Modes described in Channel Modes, exploring all the capabilities mentioned here.

These service requests originated in the TPU Enhanced Channels (either time base match, input signal transition, or link service request) result in a call to the corresponding channel service routine, which is the sequence of instructions that is called a Function. For further detail, refer to Functions.

In addition to Event Logic, each Channel has an Output Buffer Enable signal, controlled by Engine CPU, and an Enhanced Digital Filter, which eliminates spurious glitches on input pin signal. See the specific chapter information for how the Output Buffer Enable signal is controlled.

A high-level diagram of Channel logic and registers is shown in the following figure.

**Figure 126. TPU channel diagram**




### *56.5.3.1 Channel Registers and Flags*

Channel configuration and control registers can be divided in the following groups:

- Host side Configuration and Control registers.
- Engine side Configuration and Control registers.

#### 56.5.3.1.1 Host Side

The Host side register can only be accessed by the Host CPU. The Host CPU performs the following functions by configuring it:

- Host CPU can write host service requests index through any channel and send it to Engine CPU.
- Host CPU enables DMA request for any channel (triggered by events such as channel match/capture or HSR).
- Host CPU enables the interrupt request of any channel (triggered by events such as channel match/capture).
- Host CPU configuration TPU trigger output condition.
- Host CPU reads various event flags for the channel.
- Host CPU reads interrupt generation, DMA request generation, host services acknowledge flag and clear it.

Please look at the CHn_CR/SR/SCR/EFR register in the host side.

#### 56.5.3.1.2 Engine Side

The Engine side register can only be accessed by the Engine CPU. The Engine CPU performs the following functions by configuring it:

- Engine CPU controls the switch and polarity of channel output.
- Engine CPU sets the selection of entry function, the conditions of Entry point, and the priority of channel interrupt.
- Engine CPU configuration host service acknowledge index.
- Engine CPU can configure channel input filtering to be turned off.
- Engine CPU reads various event flags for the channel.
- Engine CPU reads various interrupt generation, DMA request generation, host services acknowledge flag and clear it.
- Engine CPU contains two event registers, called ER1 and ER2, which are used to configure MATCH event generation and obtain capture values.
- Engine CPU can configure channel mode, user-defined channel mode (UDCM) or predefined channel mode (PDCM).

- Engine CPU configures switches for various interrupts, DMA, MATCH, Capture and other events.
- Engine CPU configuration channel Time Base selection (TCR1 or TCR2), input capture conditions, output pin response conditions, etc.

Please look at the CHn_CR/SR/SCR/EFR/CR2/MR/ER1/ER2/ECR/OCR register in the engine side.

#### *56.5.3.2 Match Recognition*

The match operation is performed every TPU clock by comparing the channel Match1 and Match2 registers against the value of the TCR bus specified for each match. TCR1 or TCR2 bus is selected according to TBS1 and TBS2 fields. Both results have effect on the next clock cycle.

A Match 1/2 event is qualified by a set of match enabling conditions to the Chn_ER1/ER2 Registers. To recognize the match, the following match enabling conditions are required:

- For IPAC1/2=0xx, Match Enable Flag (MEF), qualified by the channel currently being serviced must be asserted. For IPAC1/2 =1xx, Match A/B is always enabled, regardless the state of the Match Enable Flag (MEF). See Chn_ER1[MEF].
- Match Recognition Latch Enable 1/2 (MRLE1/2) is asserted. A match event recognition may only occur if its corresponding MRLE1/2 bit is set, which only happens upon a write to a channel CHN_ER1/2 register by the Engine CPU. MRLE1/2 is negated when the respective match occurs or, in some double match channel modes, when a match for the other Match register occurs. It ensures that the greater-equal comparison will not cause additional matches.
- In selected modes (see Channel Modes), the particular conditions of MRL and TDL flags of the other event, i.e:
  - MRL1, TDL1 enable or block MRL2;
  - MRL2, TDL2 enable or block MRL1.
- The respective MRL is negated.
- In selected modes (see Channel Modes), the state of its respective TDL flag.

If the Match 1 and/or Match 2 conditions are met, the channel immediately forces the pin state if specified by the appropriate CHn_CR2 OPAC1/2 fields and, in some cases, by CHn_CR2 IPAC1/2 fields. Refer to Chn_CR2 Registers.

if both Match 1 and Match 2 events occur at the same time, with conflicting pin actions, the priority over the pin action is mode dependent.

##### 56.5.3.2.1 MRL1/MRL2- Match Recognition Latches

MRL1/2 indicate the recognition of a match event detected by the comparator. Assertion of MRL1/2 issues a match Service Request in specific channel modes, depending on previous events and state of SRI. After reset MRLA1 and MRL2 are both negated.

When MRL1 or MRL2 is asserted, it may change the output signal level according to CHn_CR2 OPAC1/2 fields. Assertion of MRL1/2 causes a capture of one or two time bases, according to the selected mode capturing scheme.

A match recognition is self-blocking, regardless of Channel Mode: once MRL1 (MRL2) has been asserted, it negates its associated MRLE1 (MRLE2) fields, preventing future match recognitions, until the associated match register is rewritten by Engine CPU. The Engine CPU has to enable new matches by updating the new match value in the Chn_ER1/2 register. In addition, assertion of MRL1/2 can block its twin MRL2/1, depending on the channel mode. In some double match blocking channel modes, Match1/2 event blocks the occurrence of Match 2/1 in a "first win" scheme.

It is the transition from 0 to 1 in MRL that causes the Match actions, except the service request: apart from MRLE1/2 negation(s) and service request, no action due to a Match occurs if MRL was already set to 1, even if the other MRL assert conditions are satisfied. However, if a Match and a microoperation negating its corresponding MRL occur at the same time,

MRL negation by instructions overrides its assertion, but any dependable captures and pin action occurs anyway (if MRL was already negated before), and also the negation of MRLE(s) (the respective one and, in some channel modes, the other, regardless of MRL state before). Note that MRLE must have been set before (by writing a new Match value).

#### 56.5.3.2.2 MEF – Match Enable Flag

MEF can selectively enable assertion of MRL1/2, depending on the IPAC1/2 field. For IPAC1/2=0xx, MEF=1 enables assertion of MRL1/2 for the scheduled channel during service. For IPAC1/2=1xx, Match 1/2 is always enabled, regardless the state of the MEF, but it still depends on the other Match recognition conditions. Matches of channels not being serviced are never disabled by MEF.

If a channel service needs to postpone a programmed match, MEF assures that user-code wins the race against match event after time slot transition (only for IPAC=0xx).

Note that a match event may be lost during the periods when MEF is negated only if:

- the match comparator is configured for "equal-only" behavior, and
- IPAC1/2=0xx, and
- TCR increments at the rate of TPU clock divided by 2 or faster.

When the comparator is configured as "greater-equal", the match event that occurred when MEF was negated may be recognized after MEF is asserted again, due to the "greater than" condition.

#### 56.5.3.2.3 MRLE1/2 – Match Recognition Latch Enable

MRLE1/2 is negated upon the assertion of its respective MRL1/2. In blocking match channel modes it may also be negated together with the assertion of the twin MRL2/1. The MRLE1/2 bits ensure that data captured due to the first match event will not be overwritten when MRL1/2 is negated: due to greater-equal comparison, the match condition continues to be true, but should not cause another capture event.

In addition to negation by local match event, the user-code can negate both MRLE1 and MRLE2, to block pending matches, and also MRL1/2, individually. This action will prevent future match events from the selected channel.

Writing the Chn_ER1/2 registers by Engine CPU to schedule the next match values sets MRLE1 and/or MRLE2 and enables new matches. This setting overrides the MRLE negation conditions due to channel logic or microcode (see Channel Modes). By combining write to Match 1/2 with MRLE1/2 negation microinstructions, the microcode can negate one MRLE while asserting the other.

When the match register is updated (with MRLE already asserted before) and field MRL1/2 = 1 and MRL1/2 flag is zero, the TPU behaves exactly as the TPU, that is: a match that comes concurrently with the rewrite of the match register, matching the old value, sets the MRL, as if the setting of the MRLE due to match register write had precedence over its clear by the match at that moment. After this simultaneous operation, the MRLE value stays at 1, and the captured time base value, if any, reflects the match value.

When the match register is updated (with MRLE already asserted before) and field MRL1/2 = 0 and MRL1/2 flag is zero, the match captures will occur, the MRL1/2 flag will keep negated, and MRLE will stay asserted.

### *56.5.3.3 Transition Detection and Time Base Capture*

Time Base Capture(s) occur when the value of a specified TCR is sampled into the Capture1 and/or Capture2 register. TBS1[1] and TBS2[1] select which TCR will be captured in Capture1 and Capture2, respectively.

A capture event may occur due to either of the following events:

- the assertion condition of Match Recognition Latch (MRL), even if MRL is simultaneously negated by user-code

- the assertion condition of Transition Detection Latch (TDL), even if TDL is simultaneously negated by user-code.
- any Transition Event specified by IPAC1 if both the Transition Detection Latch TDL1 and Transition Continuous Capture Enable TCCE are asserted

A capture event occurs together with the assertion of MRL/TDL, and captures the time-base value that caused the match, even if TCR1/2 increments concurrently with the assertion. MRL1/2 and TDL1 may, depending on the channel mode, inhibit the capture of the second event's TCR into Capture1/2. As a general rule, values captured by signal transitions are not overwritten by values captured by match events.

When the enable bit TCCE is asserted, captures due to Transition Events also occur after TDL1 is asserted. Those captures happen on transition events specified by IPAC1, and the TCR value is saved into Capture1 register only.

The capturing scheme is defined by the Channel Mode programmed at register field PDCM, or at register field UDCM when User Defined Channel Mode is selected. For more information on mode-dependent capture schemes refer to Channel Modes.

#### 56.5.3.3.1 TDL1/TDL2-Transition Detect Latches

TDL1/2 indicate detection of specific transition occurrences on a channel input signal. TDL1 and TDL2 assertion causes service request in single and double transition predefined modes, respectively. TDL2 assertion does not cause Service Request in single transition predefined modes, and TDL1 assertion does not cause Service request in double transition predefined modes. In single transition channel predefined modes TDL2 can be asserted on the second transition, but does not generate Service Request. Yet on predefined modes, TDL2 assertion is enabled only if TDL1 is asserted to detect an ordered input signal double transition. All the restrictions above, however, may be overridden by using the User Defined Channel Mode. The IPAC1 and IPAC2 field indicate the programmed edges of the first and second detected transition, respectively.

The sampling of a determined value (0 or 1) on the input signal due the occurrence of a Match is also treated as a "transition", depending on IPAC1/2 programming. When using a channel mode where the transition1 is initially blocked and IPAC1 is programmed to detect such "transitions", the occurrence of a Match 1 only unblocks the transition after the sampling. It means that the transition on the first Match is not effective on predefined modes where Transition 1 is enabled by Match A (m2_st, m2_dt, m2_o_st and m2_o_dt) or user-defined modes with UDCM bit M1ET asserted. A Transition 1 can still happen after the Match 1, however, if Match1 register is reprogrammed without clearing MRL1.

TDL1/2 assertion conditions initiate a capture event of one or both selected TCR buses. TDL1 or TDL2 transition event generates a Service Request, depending on channel mode, previous events and the state of SRI. For more information on service request scheme refer to Function and ISR and Channel Modes.

Assertion of TDL1/2 occurs on positive edges. The capture event occurs on the same clock, and captures the time base value present when TDL1/2 was asserted. TDL1 and TDL2 are negated during reset and may also be negated independently by microcode. TDL1/2 is reset by no way other than reset and user-code.

It is the transition from 0 to 1 in TDL that causes the Transition actions: even if TDL assert conditions are satisfied, no action due to a Transition occurs if TDL was already set to 1. However, if a Transition and a user-code negating TDLs occur at the same time and TDL was already negated, TDL negation by user-code overrides its assertion,
but any dependable captures and pin action occurs anyway.

#### 56.5.3.3.2 TCCE- Transition Continuous Capture Enable

TCCE enables capture from transitions after the TDL1 flag is set. TCCE is negated on reset, so that a capture occurs only when TDL1 asserts. TCCE can be set and reset by Engine CPU only. It can only be set together with inhibiting of the channel service requests.

When TCCE is asserted, the transition events specified by IPAC1 that occur after TDL1 is set also cause captures, into the Capture1 register only. However, output actions related to transition events are still blocked by TDL1.

#### *56.5.3.4 Channel Modes*

The Enhanced Channels support various modes of operation combining Match 1/2 recognition and transition detection events which set MRL1/2 and TDL1/2. The channel mode is individually set for each channel by TPU Engine CPU, through the PDCM field. The PDCM field selects among a set of 13 predefined channel modes, and also a user-defined channel mode.

The order in which events occur, combined with assigned channel mode, establish which following event detections are inhibited and/or enabled, as well as the actions taken: Time Base capture, flag setting (MRL1/2, TDL1/2), match disabling (MRLE1/2), output signal transition, and Service Request. Those channel mode characteristics are fixed in the predefined modes, but can be individually programmed in the user-defined channel mode.

A generic description of channel modes from the usage point of view can be found in Channel Modes Overview. Each mode is named with a mnemonic acronym for terse reference. The individual programmed attributes of the user-programmable channel mode are also described.

The modes are used differently for input and output signals, as explained in Predefined Channel Modes on Input Signal Processing and Channel Modes on Output Signal Generation. Modes also allow combining input processing and output generation in a single channel, as exemplified in Combining Input and Output Signals.

Channel Mode Logic and Event Flags shows a structural definition of channel logic and its relation to channel modes. A dynamic, event-oriented view of each channel mode can be found in Channel Mode Summary.

##### 56.5.3.4.1 Channel Mode Logic and Event Flags

Figure xxx shows a more detailed diagram of channel mode logic. The logic shown here is not necessarily identical to the actual channel logic implementation, but is equivalent with respect to conditions for event blocking, enabling, capture, and service requests.

Signals MSR, TSR, MCAP, TCAP, M1ET, M1EM2, M1BM2, M2BM1, M2BT, T1BM1, T2BM1, TBM2 and T1ET2 are decoded from programmed channel mode PDCM in predefined modes, and come directly from the UDCM register when user-defined mode is selected:

- TSR (1 bit) defines Service Requests issued by Transitions, as shown in Table 78.
- MSR (2 bits) defines Service Requests issued by Matches, as shown in Table 77.
- TCAP (1 bit) defines time base captures caused by Transitions, as shown in Table 78.
- MCAP (1bit) defines time base captures caused by Matches, as shown in Table 79.
- M1ET, M1EM2, M1BM2, M2BM1, M2BT, T1BM1, T2BM1, TBM2, T1ET2 (1 bit each) define Match and Transition reciprocal blocking and enabling, as well as Transition ordering, as shown in Table 80 and Table 81.

Table shows the decoded values of those control signals for each predefined channel mode. The first column shows the mnemonic reference for the predefined channel modes described in the following sections.

Changing PDCM or the UDCM when user mode is selected may set or reset any of the channel flags, or issue captures and service requests, so it is advisable to switch channel modes only in a quiescent channel state, with channel flags MRLE1/2, TDL1/2, MRL1/2 cleared. Furthermore, an event flag asserted in one mode may keep asserted after the mode programming change, even if the flag is impossible to be set in the new mode.

**Table 77. MSR signals - Match Service Requests**

| Value | MSR |
|---|---|
| 00 | Issue no Service Requests on Matches |
| 01 | Issue Service Request on Match 2 only |
| 10 | Issue Service Request on 2nd[1] Match |
| 11 | Issue Service Request on both Matches |

***Note:*** 1. 2nd Match means the Match that happens after the 1st Match in time, either Match 1 or Match 2.

**Table 78. TCAP and TSR signals - Transition Captures and Service Requests**

| Value | TCAP | TSR |
|---|---|---|
| 0 | 1st[1] Transition captures Time Bases corresponding to Transition 1 and Transition 2[2]<br>Transition 1 captures corresponding Time Base if it is the second transition.<br>Transition 2 captures corresponding Time Base if it is the second transition. | Issue Service Request on the 1st[1] Transition |
| 1 | Transition 1 captures corresponding Time Base.<br>Transition 2 captures corresponding Time Base. | Issue Service Request on the 2nd[3] Transition |

***Note:***

1. 1st Transition means the Transition that happens first in time, either Transition 1 or Transition 2.
2. Match capture(s) never overwrites a Transition capture. Transition captures can always overwrite a Match capture.
3. 2nd Transition means the Transition that happens second in time, either Transition 1 or Transition 2.

**Table 79. MCAP signal - Match Capture**

| Value | MCAP |
|---|---|
| 0 | Match 1 captures corresponding Time Base;<br>Match 2 captures corresponding Time Base |
| 1 | Either Match captures Time Bases corresponding to Match 1 and Match 2[1] |

***Note:*** 1. Match capture(s) never overwrites a Transition capture. Transition captures can always overwrite a Match capture.

**Table 80. TBM2 signal - Transition Blocks Match 2**

| Value | TBM2 |
|---|---|
| 0 | Transition 1 Blocks Match 2 |
| 1 | Transition 2 Blocks Match 2 |

**Table 81. M1ET, M1EM2, M1BM2, M2BM1, M2BT signals**

| Signal | Active Value Definition |
|---|---|
| M1ET (Match 1 Enable Transitions) | Transitions are initially blocked, and Match 1 enables Transitions |
| M1EM2 (Match 1 Enables Match 2) | Match 2 is initially blocked[1], and Match 1 enables Match 2 |
| M1BM2[2] (Match 1 Blocks Match 2) | Match 2 is initially enabled[1], and Match 1 blocks Match 2[3] |
| M2BM1[2] (Match 2 Blocks Match 1) | Match 1 is initially enabled, and Match 2 blocks Match 1[3] |
| M2BT (Match 2 Blocks Transitions) | Match 2 blocks Transitions |
| T1BM1 (Transition 1 Blocks Match 1) | Transition 1 blocks Match 1 |
| T2BM1 (Transition 2 Blocks Match 1) | Transition 2 blocks Match 1 |
| T1ET2 (Transition 1 Enables Transition 2) | Transition 2 is initially blocked, and Transition 1 enables Transition 2 |

***Note:***

1. The initial condition of M1EM2 prevails over M1BM2, while M1BM2 blocking prevails over M1EM2 enabling, so that Match 2 stays always blocked when both M1BM2 and M1EM2 are active. This combination is used in single-match modes (sm_*).

2. Blocking of one Match by the other is done through MRLEs.

3. Matches always block themselves by resetting their own MRLEs (Match 1 always blocks Match 1, Match 2 always blocks Match 2).

**Table 82. Predefined Channel Mode control signals decoding**

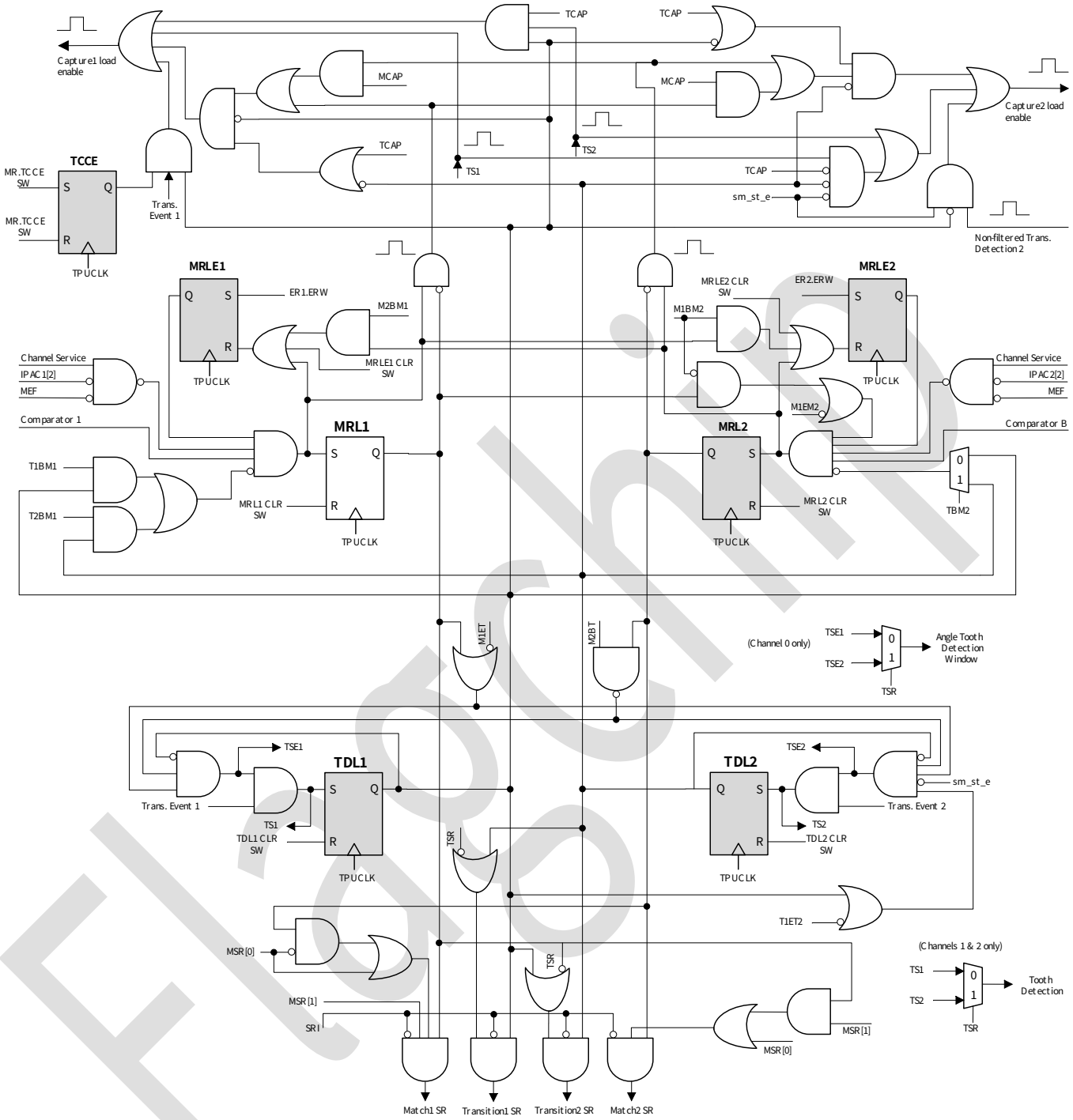
| | em_nb_st | em_nb_dt | em_b_st | em_b_dt | bm_st | bm_dt[1] | m2_st | m2_dt | m2_o_st | m2_o_dt | sm_st | sm_dt[1] | sm_st_e[2] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **MSR** | 11 | 11 | 11 | 11 | 10 | 10 | 01 | 01 | 01 | 01 | 11 | 11 | 11 |
| **MCAP** | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| **M1ET** | off | off | off | off | off | off | On | On | On | On | off | off | off |
| **M1EM2** | off | off | off | off | off | off | off | off | On | On | On | On | On |
| **M1BM2** | off | off | On | On | off | off | off | off | off | off | On | On | On |
| **M2BM1** | off | off | On | On | off | off | On | On | off | off | On | On | On |
| **M2BT** | off | off | off | off | off | off | off | off | On | On | off | off | off |
| **T1BM1** | On | On | On | On | On | Off | On | On | On | On | On | off | On |
| **T2BM1** | off | off | off | off | off | On | off | off | off | off | off | On | off |
| **TBM2** | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| **T1ET2** | On | On | On | On | On | On | On | On | On | On | On | On | On |
| **TSR** | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| **TCAP** | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

***Note:***

1. bm_dt and sm_dt are exceptions in the match blocking logic by transitions. See Both Match Request Modes (bm_st, bm_dt) and Single Match Modes (sm_st, sm_dt).

2. sm_st_e is an exception in the capture scheme. See Single Match Enhanced Mode (sm_st_e).

**Figure 127. Channel mode logic and event flags**




### 56.5.3.4.2 Channel Modes Overview

The predefined channel modes are divided, according to the way they treat transitions, in two basic modes:

- Single Transition Modes (mnemonic suffix _st): in these modes the first transition (flagged in TDL1) issues a service request, and captures both time bases (selected by TBS1[1] and TBS1[1]) except on sm_st_e. The second transition (flagged in TDL2) does not issue a service request, but it captures time base selected by TBS2[1], except on sm_st_e.
- Double Transition Modes (mnemonic suffix _dt): in these modes the second transition (flagged in TDL2) issues a service request, and each transition captures its own selected time base (Transition 1 and Transition 2 capture time bases selected by TBS1[1] and TBS2[1], respectively).

In predefined modes, Transition 2 is always (but not only) enabled by Transition 1, so that transitions are always ordered: TDL1 is set on the first transition and TDL2 on the second. Unordered transitions are possible with user-defined mode, when UDCM bit T1ET2=0. Matches are generally not ordered, except on specific ordered match modes m2_o_st and m2_o_dt. Match capture(s) never overwrites a Transition capture, while Transition captures can always overwrite a Match capture, either in predefined or user-defined modes. The following general rules apply to both predefined and user-defined modes:

- Blocking of one Match by the other, when it occurs, is done through MRLEs.
- Matches always block themselves by resetting their own MRLEs (Match 1 always blocks Match 2, Match 2 always blocks Match 1). Predefined modes differ mostly by the way matches affects and are affected by other matches and transitions, as explained in next sections. However, some general rules on Match blocking apply:
- Match 2 is blocked by first transition (TDL1) in single transition modes, and by second transition (TDL2) in double transition modes.
- Both Matches are blocked by first transition in single transition modes.

*Either Match, Blocking Modes(em_b_st,em_b_dt)*

In these modes the first match recognition that occurs blocks the other match recognition and generates a service request. They end up with one service request for two programmed match recognitions where only the first match recognition has an actual effect. If both match recognitions occur at the same time, both MRLA and MRLB are set, before the mutual blocking takes effect.

*Either Match, Non Blocking Modes(em_nb_st,em_nb_dt)*

In these modes both match recognitions are independent and each of them generates service request. Each match recognition captures its related time base and does not block the other.

*Match 2 Request Modes (m2_st, m2_dt)*

In these modes transitions are initially blocked, and are enabled by Match 1. Match 2 recognition generates the match service request and disables Match 1 recognition. Each match recognition captures its own programmed timebase. In case of simultaneous match recognition, both MRL1 and MRL2 are set, and OPAC1 field has priority over OPAC1 for selecting the pin action.

*Both Match Request Modes (bm_st, bm_dt)*

In these modes, match service request is generated only after both match recognitions occurred. By definition this is a non-blocking match mode: match recognitions do not block each other, implementing a last-served scheme. Unlike other double transition modes, bm_dt blocks Match 1 with Transition 2 (not with Transition 1), so that the second transition blocks both matches.

*Ordered Modes with Match 2 Request (m2_o_st, m2_o_dt)*

These are ordered match modes on which Match 1 recognition must precede Match 2 recognition (ordered 1->2). Match 1 asserts MRL1 and enables Match 2 and transitions. Match 2 asserts MRL2, generates a match service request, and blocks both transitions.

*Single Match Modes (sm_st, sm_dt)*

Single match modes support single or double transition with single match recognition. MRL2 is never set, and MRLE2 has no effect.

*Single Match Enhanced Modes (sm_st_e)*

This is an enhanced single transition and single capture mode, which provides nonfiltered input captures in addition to the single capture, allowing one to measure the delay of the digital filter. In an output channel, it has the same functionality of sm_st (captures both time bases at once due to a match recognition).

#### 56.5.3.4.3 Predefined Channel Modes on Input Signal Processing

When processing an input signal, the predefined channel modes can be classified in the following primary mode groups:

- Single Transition, Single Match: em_b_st, sm_st, sm_st_e
- Single Transition, Double Match: em_nb_st, bm_st, m2_st, m2_o_st
- Double Transition, Single Match: em_b_dt, sm_dt
- Double Transition, Double Match: em_nb_dt, bm_dt, m2_dt, m2_o_dt

In single transition modes, TDL1 assertion may capture both time bases at once, while in double transition modes each transition captures its related time base in its related capture register.

Double transition is always ordered, i.e TDL2 is enabled by TDL1 and generates the service request.

The channel logic supports various input modes with combinations of single/double transition and single/double match, explained in the following subsections.

*Either Match, Blocking, Single Transition(em_b_st)*

On an input signal, this mode provides double timeout mechanism on a programmed transition edge with two timebases. The signal transition blocks both pending matches, indicating that no timeout condition occurred. The two match recognitions block each other, giving good separation in the entry table as to which match recognition caused the first timeout condition, and generating only one service request. Either match performs timebase captures which do not overwrite captures by transitions.

*Either Match, Blocking, Double Transition(em_b_dt)*

In double transition mode each transition is related to one match recognition. TDL1 assertion captures its related time base, blocks Match 1 and enables TDL2. TDL2 assertion blocks Match 2, captures its related timebase and generates a service request. Match recognitions block each other, so if there is a match timeout condition on TDL1, only one match service request is generated. This mode is good for qualifying two signal transitions by match timeout mechanisms, with one service request. Note that although TDL1 assertion does not block Match 2 recognition, the value captured in Capture1 by TDL1 assertion is not overwritten by this recognition. The second transition blocks Match 2. Either match performs timebase captures which do not overwrite captures by transitions.

*Either Match, Non Blocking, Single Transition(em_nb_st)*

On an input signal, this is a double timeout mechanism of independent match recognitions of two different timebases. The match recognitions do not block each other, such that the microcode can check if one or two match recognitions occurred before their related signal transition. The signal transition detection (by IPAC1) asserts TDL1, blocks both matches, captures both time bases and generates a transition service request, indicating that none of the two timeout conditions occurred. Any combination can be easily resolved by microcode (for example, signal transition after Match 1 and before Match 2, or signal transition after both Match 1 and Match 2).

Another possible use of this mode is allocating one match recognition for transition timeout and the other for another non-critical timed task, adding functionality to a single channel. Since the transition detection blocks both match recognitions, the match recognition of the other timed task is based on the fact that the comparator checks greater-equal conditions. It may be delayed if it occurs in the period between the signal transition detection (which blocks it) and the time TDL1 is negated by user-code.

*Either Match, Non Blocking, Double Transition(em_nb_dt)*

In this mode each transition is related to one match recognition, and the match recognitions are independent of each other. This mode can be used to give independent timeout conditions for the first and the second signal transition recognitions, and call service in any case of any timeout condition.

The first transition detection programmed in IPAC1 sets TDL1, captures its related timebase, blocks Match 1 recognition and enables TDL2 assertion. The second transition detection programmed in IPAC2 sets TDL2, blocks Match 2 recognition, captures its related timebase and generates a service request. Any match recognition that occurs captures its related time base and generates a match service request, independent on the other match recognition.

*Match 2 Request, Single Transition(m2_st)*

On an input signal, this mode provides an open window filter for a single signal transition. MRL1 assertion opens the window, and enables transition detection on TDL1 from this time on. MRL2 assertion blocks Match 1 (by negating MRLE1), providing conditional window opening, because transitions are indirectly blocked. It also generates service request, but if it happens after Match 1 it does not block transitions, providing a non-blocking timeout mechanism for the estimated signal transition time (typically it indicates a missing transition, or mis-prediction of the transition time).

Transitions can be detected from the TPU clock following MRL1 assertion. The Transition 1 detection asserts TDL1, blocks both matches, captures both timebases and generates service request.

Using this mode, the channel can replace software open window filtering of qualified transitions with the channel hardware window. The window opening and timeout can be scheduled for any of the two time bases or combination of them. Typically, Match 1 will be used to open a prediction window, and Match 2 will be used as a timeout condition which does not close the prediction window. This configuration improves noise immunity from early signal transitions, and reduces the probability for blocking late signal transitions due to timeout mis-prediction.

Using these conditions, the microcode can easily resolve the state:

- If TDL1 and MRL2 are asserted and MRLB negated, signal transition is in the expected range.
- If MRL1 and MRL2 are both asserted, and TDLA1 is asserted, the signal transition had a timeout condition due to Match 2 mis-prediction.
- If MRL2 is asserted and TDL1 negated, a timeout condition occurred, and the expected signal transition had not occurred yet.
- If MRL1 is negated and MRL2 is asserted, the conditional window did not open at all (for example: a time window is open only after a specific angle, otherwise it is not opened).

*Match 2 Request, Double Transition(m2_dt)*

This mode is used as an open window filter for two signal transitions. In this case the Match 1 recognition opens the window (unless Match 2 recognition occurred first), and Match 2 recognition blocks Match 1 and generates a match service request. It is similar to m2_st, but in this case, it is the second transition that blocks Match 2. MRL2 assertion is a global timeout condition for the two pulses. Like m2_st, MRLB can conditionally eliminate the window from opening.

Using the TDL1, TDL2, MRL1 and MRL2 conditions, the microcode can easily resolve the state, in a similar manner as m2_st, with additional information on the second transition (TDL2).

*Both Match Request, Single Transition(bm_st)*

On an input signal, this is a double timeout mechanism on two different time bases. Both match recognitions must occur before the signal transition to generate a match timeout service request. Assertion of TDL1 blocks both Match 1 and Match 2 recognitions, and captures both time bases, indicating there was no double timeout condition from both time bases.

Using the same timebase implements two timeout conditions, the first only sets its related MRL and the second generates a service request. Using these flags allows the microcode to check if one or both match recognitions precede the signal transition.

### *Both Match Request, Double Transition(bm_dt)*

In this mode the first transition detection does not block matches, since both match recognitions are required to generate a match service request. The second transition detection asserts TDL2, blocks Match 1 and Match 2, captures its related timebase and generates transition service request. In this mode, a Match A recognition which occurs after the assertion of TDL1 does not capture a new value in Capture1, to preserve the actual signal transition time. Assertion of TDL1, however, always captures its related timebase.

This mode allows putting a double match timeout condition on the second transition. Typically, a pulse trailing edge timing can be checked against two time bases, to indicate if the pulse has not ended when both MRLA and MRL2 are asserted. When a transition service request is generated by TDL2 assertion, the state of MRL1 and MRL2 indicates which timeout condition occurred, if any.

### *Ordered Modes with Match 2 Request, Single Transition (m2_o_st)*

On an input channel, this mode provides a closing window filter for a single signal transition. Match 1 assertion captures its programmed time base in Capture1, opens the filter window (enables assertion of TDL1), and enables assertion of MRL2. Match 2 recognition captures its related timebase, closes the window (disables assertion of TDL1) and generates a service request. Due to Match 1 and Match 2 ordering, the window is opened for at least one microcycle. Match 2 recognition indicates a window timeout condition which blocks late signal transitions, outside the prediction window. Transition detection blocks both matches, indicating the transition occurred inside the estimated window. Transitions can be detected from the TPU Clock following MRLA assertion until the TPU clock on which MRL2 is asserted. When TDL1 is asserted inside the window range it disables both matches, captures both time bases and generates a transition service request.

Using this mode, the channel can replace software window filtering of qualified transitions with the channel hardware window. The window opening and closing time can be scheduled for any of the two time bases or a combination of them.

### *Ordered Modes with Match 2 Request, Double Transition (m2_o_dt)*

In this mode the channel logic implements a window filter for two detected signal transitions. MRL1 assertion captures its related timebase and enables the assertion of both TDL1 and TDL2. MRL2 assertion captures its related timebase and disables the assertion of both TDL1 and TDL2. Transitions can be detected from the TPU Clock following MRL1 assertion until the TPU CLock on which MRL2 is asserted. The first signal transition (following MRL1 assertion) asserts TDL1, captures its related timebase and enables assertion of TDL2. The second signal transition detection asserts TDL2, blocks Match 2, captures its related timebase and generates the service request.

If both signal transitions occur inside the scheduled window, Match 2 recognition is blocked. If one or both signal transitions do not occur inside the scheduled window, Match 2 recognition generates a match service request and blocks further transition detections. The microcode can resolve the state using MRL1, MRL2, TDL1 and TDL2, which affect the entry point selection.

### *Single Match Enhanced Mode(sm_st_e)*

This is an enhanced single transition and single match channel mode which provides timing information of the digital filter delay.

The Capture1 register captures the timebase selected by TBS1 due to transition detection specified by IPAC1 or match recognition, as in sm_st mode. Initially, the Capture2 register continuously captures the unfiltered IPAC2-selected signal

transitions from the digital filter input, directly from the signal synchronizer. When an IPAC1-qualified, filtered transition detection occurs, TDL1 is set, MRL1 assertion is blocked, and, in addition, captures into Capture2 are also blocked.

*Single Match, Single Transition(sm_st)*

In this mode the channel logic is functionally back-compatible to a TPU single action channel, but a match or transition detection captures at once both timebases. The mode recognizes a single transition with single match timeout. Either TDL1 or MRL1 generates service request and captures both timebases. Assertion of TDL1 blocks future assertions of MRL1.

*Single Match, Double Transition(sm_dt)*

In this mode, the first transition detection asserts TDL1, captures a timebase in Capture1 and enables TDL2. The second signal transition asserts TDL2, blocks Match 1, captures a timebase in Capture2 and generates a service request.

Match 1 (before TDL2) captures into Capture2 the timebase selected by TBS1, in order not to overwrite the captured value of TDL1.

This mode is used for scheduling one timeout condition on two input signal transitions (pulse timeout).

#### 56.5.3.4.4 Channel Modes on Output Signal Generation

Since channel logic can generate output signal transitions based on Matches, the channel can be viewed as working in the following primary mode groups for signal generation:

- Single Match: em_b_st, sm_st, sm_st_e, em_b_dt, sm_dt
- Double Match: em_nb_st, bm_st, m2_st, m2_o_st, em_nb_dt, bm_dt, m2_dt, m2_o_dt

The channel logic supports various match channel modes with single/double match, as explained in the following subsections.

*Either Match, Blocking Modes(em_b_st,em_b_dt)*

On an output signal these modes are useful when using two different time bases to set a required signal transition. The first match condition which is met sets a required pin action, captures both time bases, blocks any effects of the other recognition, and generates a service request. Because the first match recognition blocks the other, the user-code can get good separation in the function entry table as to which match caused the timeout first, and both time bases are captured, enabling the microcode to compare one timebase to the other at the moment of the match recognition. These modes can be used for:

- Scheduling a required pin action to the first match recognition of two different time bases.
- Canceling a programmed pin action scheduled on one time base by match on another timebase. User-code has to set the OPAC field of the canceling match to no-action and the OPAC field of the other match to the required pin action which may be blocked. If Match 1 is the canceling match, it blocks the pin action also in case of two matches at the same time, since it has priority in this case. If Match 2 is the canceling match, it does not block the pin action in case of two matches at the same time.

*Either Match, Non Blocking Modes(em_nb_st,em_nb_dt)*

On an output signal these modes are useful in combination with the set on the entry point, to define an interlaced operation. For example, each match recognition can set a pin action, and the second pin action is not sensitive to microcode latency. Example for usage is PWM interlaced function on which the latency is determined by the period and not the duty cycle.

Another possibility is using one match for pin actions and the other match for an unrelated timed task without pin action (double the functionality of a single channel).

*Match 2 Request Modes (m2_st, m2_dt)*

On an ouput signal, these modes can generate narrow pulses or do conditional pin actions. A conditional pin action means that the pin state is changed only if the match recognitions occurred in the correct order, since the Match 2 recognition which generates the service request also has priority over the pin action and blocks future Match 1 recognitions.

Setting OPAC1 to a desired pin action and OPAC2 to no-action, and using different time bases for Match 1 and Match 2 defines a conditional OPAC1 pin action which can be blocked by Match 2 recognition. For example, setting Match 1 on time and Match 2 on angle can limit the pin action to a maximum angle value. When pulses are generated, the service is requested at the trailing edge of the pulse, after MRL2 is asserted.

*Both Match Request Modes (bm_st, bm_dt)*

On an output signal, each match recognition can affect the pin state, and capture its programmed time base. This way the pin action can be programmed separately for both match recognitions. For example, both match recognitions can negate the signal, and service request is generated after both conditions are met. This mechanism can set two conditions to do a required pin action, and the first recognition changes the signal, but service is called only after both conditions occur.

When using the same time base, these modes can generate narrow pulses in any required order. For example, in a PWM function, when duty cycle is below 50% the function can get service on the low time and program the pulse to the required duty cycle of the high time. When duty cycle is equal or above 50%, the function can get service on the high time and program a negative pulse with the width of the required low time. To switch between the two states the function can program once the same transition time to Match1 and Match2 with a required pin action, and on the next service program double match for the new state.

Another usage is generating a required pin action on one programmed time and service request later on another time, after the second match recognition occurs, or capturing some timebase on one time and generating a required signal transition and service request later.

*Ordered Modes with Match 2 Request (m2_o_st, m2_o_dt)*

The order of the match recognitions imply that OPAC1 field programmed pin action always precede the OPAC2 register pin action. Setting OPAC1 to no-action, based on the greater-equal comparator, enables using Match 1 on one time base to delay the signal effect of Match 2 on the other time base. This method implements a conditional pulse extension or conditional delay on signal transition. These modes can also be used for deferred pulse generation with microcode service request after its trailing edge (if Match 1 condition comes after Match 2 condition).

Another option is having Match 1 recognition associated with output pin actions and Match 2 recognition for a timed microcode task which has to be scheduled at a programmed time which may be delayed by the Match 1 pin action.

*Single Match Modes (sm_st, sm_dt)*

There is no difference between plain and enhanced Single Match modes on an output signal.

In this mode the channel logic is functionally back-compatible to a TPU single match output channel. Match 1 recognition generates service request and sets the pin state according to OPAC1 field. It captures at once the timebase selected by TBS1 in Capture1 and the timebase selected by TBS2 in Capture2.

*Match/Transition Pin Action Confict Resolution*

In output signals, matches and/or transitions automatically cause pin actions defined by the OPAC1/2 and/or IPAC1/2 fields. Simultaneous matches/transitions may be associated with different, possibly contradictory, pin actions. These conflicts are resolved according to the table below.

If an OPAC1/2=000 (no action) prevails over non-zero OPAC according to the table below, then if Match 1/Transition 1 and Match 2/Transition 2 occur simultaneously, no output pin action occurs, that is: a match on the action logic with OPAC=000 inhibits simultaneous actions of the other OPAC, if prevailing according to the following table.
That also applies when output actions are caused by inputs (OPAC=1xx).

**Table 83. Simultaneous match pin action priority**

| Channel Mode | Priority |
|---|---|
| em_nb_st/ em_nb_dt | OPAC1 |
| em_b_st / em_b_dt | OPAC1 |
| bm_st / bm_dt | OPAC2 |
| m2_st / m2_dt | OPAC2 |
| m2_o_st/m2_o_dt | In these modes, there is no possibility of simultaneous matches |
| user-defined | OPAC2 if M2BM1 = 1 and M1BM2 = 0 OPAC1 otherwise |

#### 56.5.3.4.5 Combining Input and Output Signals

The processing of input signal can be combined with output signal generation. A detected input transition can trigger an output signal edge, even without microcode intervention, by using OPAC options 1xx.

The channel set-up examples below show these two capabilities combined (see the figure below).

The first example implements a fast short-circuit protection feedback mechanism for driving high-current output devices. The signal after the high-current driver feeds back to the channel input. The input signal is normally delayed from the output signal by the device turn-on delay. After the channel output turns on, the channel logic must check if the driver output (connected to the channel input) follows the driven value after the maximum device turn-on delay. If it does not, the driver output is probably shorted, and the channel output must be turned off immediately to avoid damaging the device.

Match 1 causes the output to be driven high (for simplicity the output and input signals are shown as positive logic). It also causes a transition 1, because IPAC1=100 and the input is still low. Match 2 occurs after the expected driver delay, and causes a service request. If the output is shorted, a Transition 2 occurs on Match 2 because IPAC2=100. This will cause the output to go low immediately, because OPAC2=100.

In the second example, an output pulse is generated from an input transition without user-code intervention. Match 1 opens a window for transitions and also enables Match 2. A rising edge on input sets output high. On Match 2 the window closes, and input signal is checked: if sampled high, the output resets; otherwise it stays high.

In the third example, a pulse is generated depending on the value sampled on the input signal at a predetermined time. Match 1 samples the input signal, causing a Transition. Low level on input sets output low, otherwise it stays high. Match 2 sets output high. In both cases a service request is issued (microcode intervention), at the beginning and at the end of the pulse (Match 2), if required (SRI = 0).

**Example 1: Short-circuit protection feedback**

IPAC1 := 100; OPAC1 := 001; Match1 := output activate time;

IPAC2 := 100; OPAC2 := 100; Match2 := Match1 + max. high-current driver turn-on delay;

PDCM := bm_dt;

**Example 2: Pulse generation on windowed input transition**

IPAC1 := 001; OPAC1 := 101; Match1 := window open time;

IPAC2 := 101; OPAC2 := 100; Match2 := window close time, input sampling;

PDCM := m2_o_dt;

**Example 3: Pulse generation on input sampling**

IPAC1 := 100; OPAC1 := 100; Match1 := window open time, input sampling;

IPAC2 := 000; OPAC2 := 001; Match2 := window close time = Match1 + pulse width;

PDCM := em_nb_dt;

#### *56.5.3.5 Enhanced Digital Filter-EDF*

The EDF eliminates passing of signal transitions which are caused by noise. Its purpose is to eliminate false transition service requests caused by noise pulses which are shorter than a programmed width.

See the chip-specific TPU information for connections between TPU and input glitch filters.

The EDF has three modes of operations, selected by the CDFC field in the GCR_CR register (see the GCR_CR register in Engine side). These modes offer selections of trade-off between noise immunity and signal latency. CDFC also allows the filter to be bypassed. Table 84. Pulse widths and delays gives an example of minimum detected signal pulse and maximum filtered noise pulse in the three EDF operation modes. In Angle Mode, if AM=01, the EDF in channel 0 is replaced with the digital filter and synchronizer of the TCRCLK signal. In this mode, channel 0 works in combination with the Angle Counter logic, and their operation is fully synchronized.

Following subsections provide the functional description of the TPU channel digital filter.

##### 56.5.3.5.1 Two-Sample Mode

It uses the filter clock which is the clock divided by (2, 4, 8,.., 256) as a sampling clock. The filter clock is selected by the FPSCK field in the GCR_CR in Engine side. The EDF compares two consecutive samples. If both samples have the same value, the input signal state is updated. Note that when the FPSCK field selects the TPU clock divided by two, the EDF works like the TPU four-clock digital filter.

##### 56.5.3.5.2 Three-Sample Mode

In this mode, the EDF uses the filter clock as a sampling clock. The EDF compares three consecutive samples. If all three samples have the same value, the input signal state is updated.

The Three-Sample mode gives more signal latency than the Two-Sample mode, but also better noise immunity and better ratio between minimum detected signal pulse to maximum filtered noise pulse. When a certain filter clock frequency is selected for Two-sample mode, double filter clock frequency can be selected to get better latency in Three-sample mode.

##### 56.5.3.5.3 Continuous Mode

In this mode the EDF compares all the values sampled at the rate of TPU clock divided by two, between two consecutive filter clock pulses. If the signal is continuously stable for the entire digital filter clock period, the input signal state is updated.

This method gives the same latency and the same ratio between minimum detected signal pulse to maximum filtered noise pulse, as the Two-Sample mode, as long as there is no noise. Each sampled noise delays the signal transition detection by at least a whole digital filter clock period.

The Continuous mode gives the best noise immunity by comparing multiple samples of the noise. On the other hand, when a short noise pulse appears in the middle of the filter clock period at the same time of a real signal transition, the Continuous mode may reject a real signal transition and delay the response to the first filter clock period in which the signal is continuously stable. This may add to the latency and also to the minimum detected signal pulse in a noisy environment.

#### 56.5.3.5.4 Bypass Mode

In bypass mode the signal that feeds the edge detection comes directly from the output of the synchronizer, not filtered.

#### 56.5.3.5.5 Filter Clock Prescaler

Each channel configured as an input has an associated synchronizer followed by a digital filter connected to the signal that samples signal transitions. After reset, the digital filter filters out high and low pulse widths smaller than the period of two TPU clocks with GCR_CR bit FCSS=0, or 1 TPU clock with FCSS=1, preventing these transitions from being input to the transition detect logic. For FPSCK=0 and FCSS=0, the synchronizer and digital filter are guaranteed to pass pulses that are as wide as or wider than four TPU clocks, meaning a minimum period of 8 TPU clocks. These figures are halved by setting FCSS=1. By changing the FPSCK field in register GCR_CR the user can select a lower clock rate for the filter signal to define wider valid pulses and filter out wider noise pulses. The filter prescaler clock control is a division of the TPU clock. To guarantee pulse detection by the digital filter, the pulse must cover at least the stated number of samples at the filter clock rate. For example, a two-sample digital filter must sample two points in the pulse to detect it. The table below shows the minimum guaranteed detected pulse width and the maximum filtered noise pulse width. The table refers only to the digital filter operation. The external pulses may have to be wider (to ensure detection) or narrower (to ensure filtering) depending on the rise/fall delay differences in the MCU receivers and internal logic. Delays introduced by synchronizer, filter and edge detection logic are explained in Input/Output Signal Delays.

The TCRCLK signal also has a synchronizer and a filter. The FCSS bit has no influence on the clock prescaler of the TCRCLK filter, which works like the channel filter with FCSS=0.

**Table 84. Pulse widths and delays**

| Filter Control (FPSCK) | | Filter Clock Period[1] | Min. width guaranteed detected / max. width filtered (min. filter delay / max. filter delay)[1] | |
|---|---|---|---|---|
| FCSS = 0 | FCSS = 1 | | Two-Sample or Continuous mode | Three-Sample or Integrator[2] mode |
| N/A | 000 | 1 | 2 / 1 (2 / 3) | 3 / 2 (3 / 4) |
| 000[3] | 001 | 2 | 4 / 2 (3 / 3) | 6 / 4 (5 / 5) |
| 001 | 010 | 4 | 8 / 4 (5 / 7) | 12 / 8 (9 / 11) |
| 010 | 011 | 8 | 16 / 8 (9 / 15) | 24 / 16 (17 / 23) |
| 011 | 100 | 16 | 32 / 16 (17 / 31) | 48 / 32 (33 / 47) |
| 100 | 101 | 32 | 64 / 32 (33 / 63) | 96 / 64 (65 / 95) |
| 101 | 110 | 64 | 128 / 64 (65 / 127) | 192 / 128 (129 / 191) |
| 110 | 111 | 128 | 256 / 128 (129 / 255) | 384 / 256 (257 / 383) |
| 111 | N/A | 256 | 512 / 256 (257 / 511) | 768 / 512 (513 / 767) |

***Note:***

1. This table shows pulse widths and delays in number of periods of the TPU clock.
2. Integrator mode is available for TCRCLK filtering only, see TCRCLK digital filter.
3. These delays also apply to the TCRCLK filter when TCRCF selects TPU clock/2 as filter clock.

### 56.5.4 Time Base

TPU has two Time Counter Registers, TCR1 and TCR2. They provide 24-bit time bases, shared by all 32 channels. Any channel can use both time bases to:

- Match channel's internal registers Match1 or Match2;
- Capture time base value to channel's internal registers, Capture1 and/or Capture2, when a match recognition or an Input transition detection occurs. For more information on channel events refer to Enhanced Channels.

The TCR1 and TCR2 counters are accessible by the Engine CPU for read and write operations. Its current value is used for getting the current time, and the captured values are used for channel relative time calculations of future events. When they are written at the same time they are incremented from any clock source, the written value prevails.
TCR1 with TBR_CR[TCR1CTL]=0 and TCR2 values are updated. TCR1 can also work at full-speed TPU clock, and so be updated, when TBR_CR[TCR1CTL]=1.

The TCR2 counters between the two Engines are out of phase by 1 TPU clock, even when Time Bases are shared between them through FCB BUS. It also applies to TCR1 counters if TBR_CR[TCR1CTL]=0, but they can be in phase otherwise.

#### *56.5.4.1 Timer Count Register 1 - TCR1*

TCR1 can be used in the following modes:

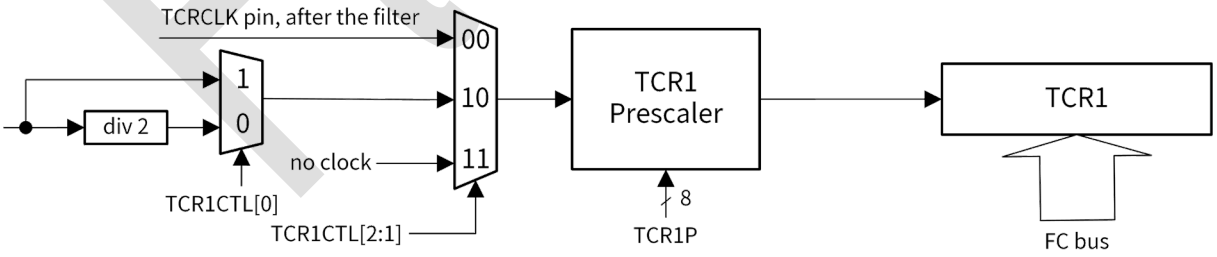
- Internally Clocked Mode
- Externally Clocked Mode

The host program can read TCR1 time base through the TBR_T1R (see the TBR_T1R register). The TCR1 bus runs through all the local engine channels. In channels which select TCR1 as Match1 and/or Match2 source, when its value is greater or equal to the programmed match value, a Match 1 and/or Match 2 event occurs on that channel. A recognized match event sets its related Match Recognition Latch 1 or 2, and according to the Predefined Channel Modes (PDCM) it may generate a channel service request. For details on TPU channels refer to Enhanced Channels.

##### 56.5.4.1.1 Externally Clocked Mode

TCR1 can be driven externally by the TCRCLK input, after the digital filter. The TCR1 clock source is configured by the TCR1CTL bit, as shown in the following figure. For more information on clock source selection, please refer to the TBR_CR register.

**Figure 128. TCR1 clock selection**




##### 56.5.4.1.2 Internally Clocked Mode

TCR1 can be driven by the TPU clock or TPU clock divided by 2, before the prescaler. TCR1CTL can also be used to freeze TCR1 clock independently of TCR2.

##### 56.5.4.1.3 TCR1 Clock Prescaling

Any clock source selected by TCR1CTL is prescaled by a factor of 1 to 256, selected by TBR_CR field TCR1P. For more information on prescaler configuration refer to the TBR_CR register.

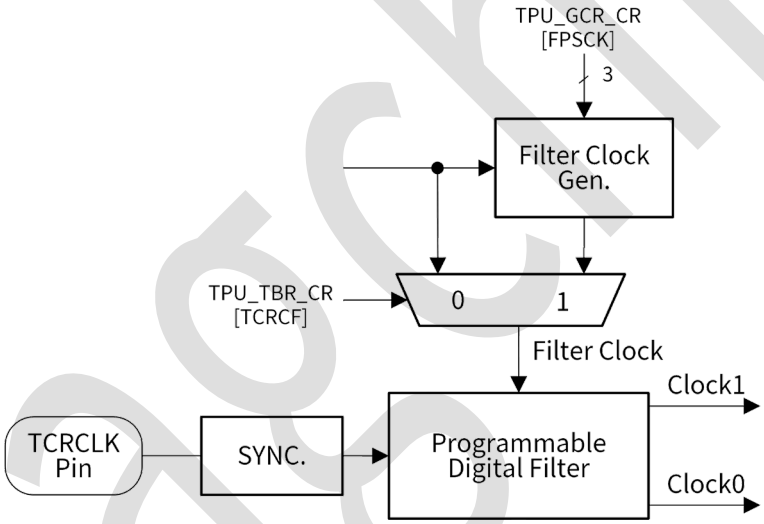
#### 56.5.4.2 Timer Count Register 2 - TCR2

The TCR2 is a 24-bit counter which can be used in the following modes:

- Pin Transition Mode: Count the rise, fall or both transitions of TCRCLK signal.
- Angle Clock Mode: Count internal tooth angle in combination with the TPU Angle Counter (EAC) hardware which implements an Angle PLL, and generates angle information to the channels. This mode is targeted for angle-based applications.
- Gated Mode: Count with rate derived from the TPU clock divided by eight. The TCRCLK signal is used to gate this count, enabling pulse accumulator operations.
- Internally Clocked Modes: TCR2 is driven by internal clock, with count rate of TPU clock divided by eight.

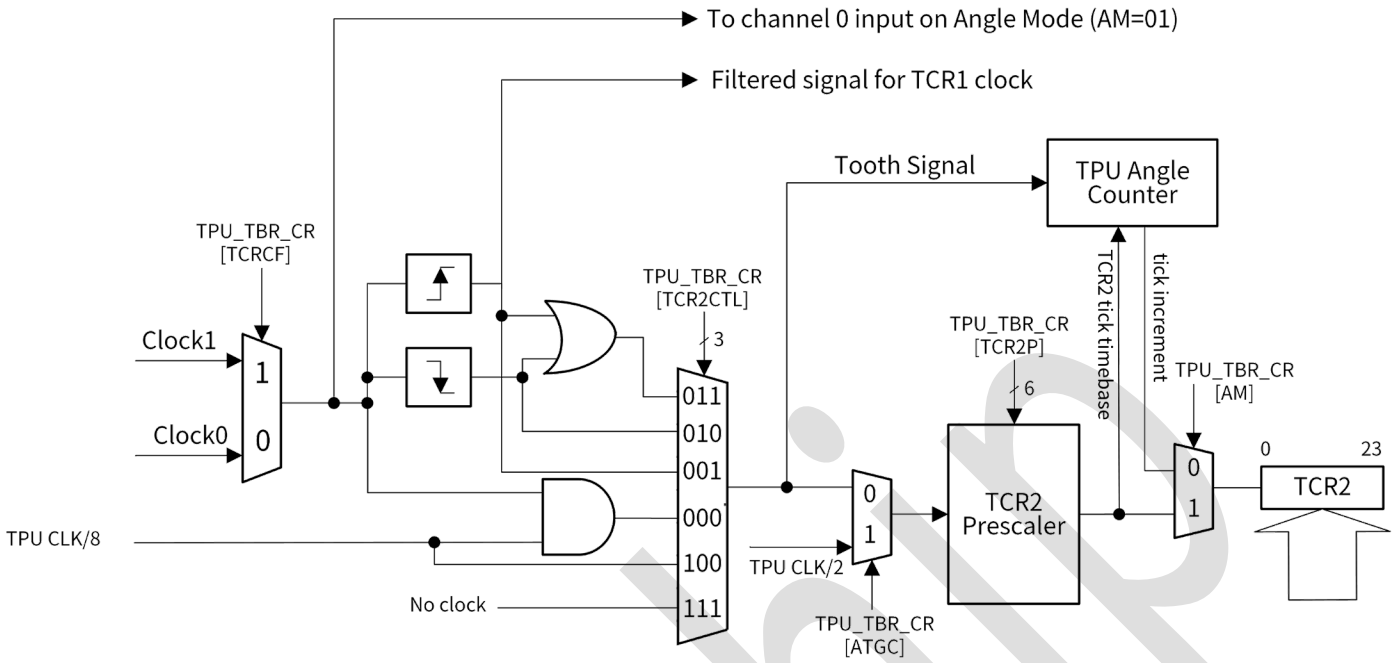
All clock sources pass through a prescaler. In addition, the TCR2 count can be originated from the EAC which is a hardware angle clock and angle counter. Figure 129. TCR2 filter and Figure 130. TCR 2 prescaler show the diagrams for TCR2 clock control. When TCR2 is not driven by the EAC, the TBR_CR field TCR2CTL selects the clock source, also allowing TCR2 to be frozen independently of TCR1 (see the TBR_CR register). When in Angle Mode, TCR2CTL selects the TCRCLK edge sensitivity.

**Figure 129. TCR2 filter**




Flagchip

**Figure 130. TCR2 prescaler**




The TCRCLK signal input is passed through a synchronizer and a programmable digital filter. In Angle Mode with AM=01, synchronizer and filter are also used in Channel 0, replacing its input synchronizer and filter, to get the same timing in the EAC and Channel 0. The TCRCLK synchronizer is an improved filter that provides best latency while maintaining proper noise filtering (see the field TCRCF in TPU TCR_CR Register.

The TCR2 bus runs through all the local engine channels. In channels which select TCR2 as Match1 and/or Match2 source, TCR2 value is compared against Match1 and/or Match2 registers. A recognized match event sets its related Match Recognition Latch 1 or 2, and according to the Predefined Channel Modes (PDCM) it may generate a channel service request. For details on TPU channels refer to Enhanced Channels.

The TCR2 counter is accessible by the Engine CPU for read and write operations. Its current value is used for getting the current counter value (representing signal transitions, time or angle), and the captured values are used for channel relative count calculations of future events. The TCR2 value is readable to the host through the TBR_T2R register.

#### 56.5.4.2.1 TCR2 Clock Prescalling

Except in Angle Mode, any clock source selected by TCR2CTL is prescaled by a factor of 1 to 64, selected by TBR_CR field TCR2P. For more information on prescaler configuration refer to the TBR_CR register. When TCR2 is written by Engin CPU, the prescaler is reloaded with TCR2P and it becomes effective, if the TPU GTBE input is asserted.

The counter that divides the TPU clock by 8 before the prescaler also resets when the TPU GTBE input is negated, or when TCR2 is written by Engine CPU.

In Angle Mode the TCR2 prescaler is not used to directly increment TCR2, but can be used, selected by TBR_CR bit ATGC=1, as a clock source for the EAC generation of the angle ticks which increments TCR2. In this case, the TCR2 prescaler input is TPU clock/2, independently of TCR2CTL. See Generating the Angle Ticks for details.

#### 56.5.4.2.2 TCR2 Gated Mode

TCR2 Gated mode is selected in field TCR2CTL of register ETPUTBCR. In this mode the TCRCLK signal enables or disables transfer of the TPU clock divided by 8 to the TCR2 prescaler. By programming the prescaler, TCR2 can run at rates from TPU

clock divided by eight down to TPU clock divided by 512, in steps of eight TPU clock divisions. For more information refer to the TBR_CR register.

#### 56.5.4.2.3 TCR2 Signal Transition Modes

These modes are selected when the TCR2CTL field in TBR_CR is set to rise, fall or "rise-and-fall". In these modes the TCRCLK signal is the TCR2 clock source, and its maximum transition rate depends on the TCRCLK digital filter mode of operation. The TCRCLK digital filter can be programmed to use the TPU clock divided by two, or use the same filter clock of the channels, controlled by the TCRCF field in TBR_CR. It contains an up-down counter which operates as a digital integrator, optimizing signal latency in the selected mode and clock rate.

When TPU clock divided by two is selected, the synchronizer and the digital filter are guaranteed to pass pulses that are wider than four TPU clocks (two filter clocks). Otherwise the TCRCLK is filtered with the same filter clock as the channel input signals. For details on TCRCLK and channels digital filter control refer to the TBR_CR register and Enhanced Digital Filter - EDF.

#### 56.5.4.2.4 TCR2 Bus in Angle Clock Mode

In this mode the TCR2 counter operates as part of the TPU Angle Counter (EAC). The TCR2 bus value reflects this angle representation in which it counts Angle Ticks. Angle Mode is selected when the AM bit is set in TBR_CR.

Note that when TCR2 works in Angle Mode, it does not count directly from the TCR2 clock input which indicates tooth signal transition. Its Angle counter is controlled by the Count Control and High Rate logic (see Count Control and High Rate Logic), which provides the interpolated pin position, and handle cases of missing tooth, acceleration, de-acceleration and mechanical corrections.

The EAC uses the TCRCLK signal to get the tooth transition indications. The TCR2CTL field in TBR_CR has to be set for the appropriate tooth edge detection rise, fall, "rise-and-fall" or none. TCR2 count clock comes from the EAC control and not directly from the physical tooth. This way the EAC control processes the signal transitions and handles missing teeth and flywheel mechanical corrections. Note that when TCR2CTL selects "none" for tooth edge selection, the TCR2 is not necessarily frozen, but can still be incremented by the EAC logic.

In Angle Mode, TPU channel 0, 1 or 2 operation is combined with the EAC operation. When channel 0 is selected for EAC operation, the TCRCLK digital filter is used both by the EAC and by channel 0 to get full synchronization between the two logics.

The TPU Angle Counter (EAC) logic runs continuously and updates the TCR2 Angle counter, eliminating the microcode latency in updating the TCR2 value.

### *56.5.4.3 TCRCLK Digital Filter*

The TCRCLK signal has an improved integrating digital filter with a 2-bit up-down counter. The counter counts up to 3 when a high signal level is detected, or down to 0 when a low level is detected. The signal state is updated to one when the counter stops at 3, or zero when the counter stops at 0. The field TCRCF in TBR_CR register (see the TBR_CR register) determines whether the TCRCLK signal input (after a synchronizer) is filtered with the same filter clock as the channel input signals (see Enhanced Digital Filter - EDF) or uses the TPU clock divided by 2, and also whether the TCRCLK digital filter works in integrator mode or the same two sample mode as the channel filters.

The TCRCLK filter delay and prescaling determines the minimum detectable TCRCLK pulse widths and, therefore, its maximum frequency, as shown in Filter Clock Prescaler and Table 84. Pulse widths and delays. The TCRCLK signal delay from the module input to TCR1/TCR2 incrementing or detection in the EAC logic is explained in Input/Output Signal Delays.

### 56.5.5 EAC-TPU Angle Counter

#### *56.5.5.1 General*

The EAC logic contains a mechanism which follows the flywheel angle, based on the tooth rate. This hardware works in combination with the TCRCLK signal, the TCR2 counter and Channel 0, 1 or 2 (depending on the TBR_CR field AM) to generate angle information on the TCR2 bus which is passed to all the local engine channels. The EAC helps to implement a digital angle PLL (see Figure 134. EAC PLL), which combines hardware with Engine CPU processing at channel 0, 1 or 2. The angle measurement is based on history knowledge of the tooth period, for predicting the period of the next tooth. The tooth period is partitioned into a programmable number of Angle Ticks. The TPU application will use the divider in the Engine CPU ALU unit to calculate an integer and a fraction part of the angle tick such that the full tooth period gets the correct programmed number of angle ticks with no accumulated error.

Each single tooth can be divided in angle ticks, up to 1024. In a 60 teeth flywheel, 128 Angle Ticks per tooth provide resolution of ~0.05 degrees per tick, which meets the accuracy requirement of 0.1 degrees in current automotive applications.

The measurement of one tooth in angle ticks is independent on engine RPM; it is the tooth period itself (and the corresponding tick period) that is re-calculated for each new tooth, based on the difference between the estimated tooth and the actual detection.

For these applications, one of the TPU channels 0, 1 or 2 is dedicated to service the physical tooth detection. Channel 0 shares the same filtered input as the TCRCLK signal to get the same timing as the EAC. The TCRCLK edge detection is selected by TBR_CR field TCR2CTL for the EAC, and by IPAC1/2 on channel 0, which must be set to detect the same edge(s). When channels 1 or 2 are selected to work with the EAC, IPAC1/2 is used to select the tooth signal edge detection for both the channel and the EAC, and the tooth signal that feeds the EAC is the same filtered input which feeds the channel.

Channel 0, 1 or 2 generates the signal transition service request, and can also be used for generation of a window filter on this transition, to qualify TCR2 clocks. For this purpose, the selected channel should be configured with double match window filtering mode (refer to Channel Modes). Depending on the channel mode set for the channel, Match 1 recognition opens the window, and Match 2 recognition may close it or leave it open. See Angle Logic and Channel Modes for details. Match 2 also generates a time-out service request. Its input signal transition comes from the tooth. The window can be defined by Engine CPU to open at a predefined point inside the tooth period, and stay open for a desired percentage to the tooth period. The window can be measured in angle or time This method improves the noise immunity by allowing transition detection only on an expected period.

The EAC supports deceleration, acceleration, last tooth and missing tooth scenarios. The large range of angle ticks per tooth can be used to cover longer tick counts caused by one or more missing teeth, or to provide extra resolution for future application requirements. In case of a missing tooth, the EAC can be configured to insert a dummy tooth or to simply measure a longer tooth.

Figure 135. Angle counter system shows the block diagram of the Angle Counter system. TCR1 or TCR2 prescaler is used as a time base which measures the tooth period and is used for partitioning the period to angle ticks.

#### *56.5.5.2 Angle Mode Register*

In Angle Mode, the registers described below control TPU angle operations. They are accessible only by Engine CPU as source and destination registers in microinstructions. When TPU is not in angle mode (AM bit is negated in TBR_CR register), all angle mode registers can be used as general purpose registers.

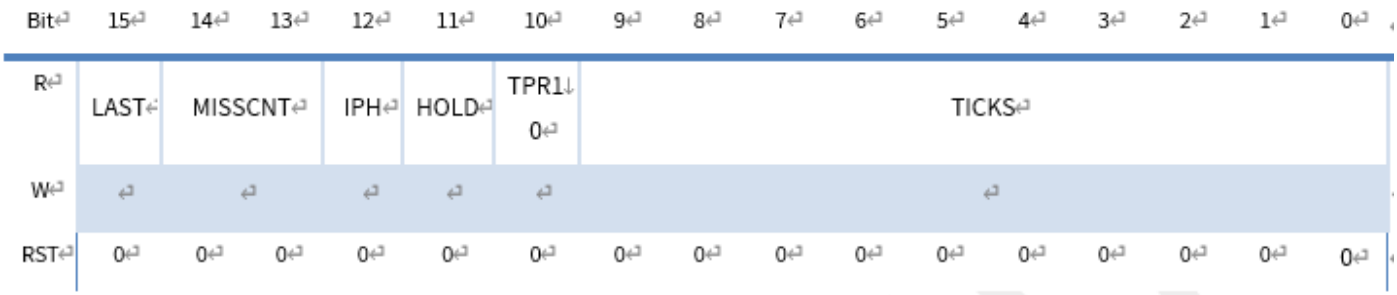
#### 56.5.5.2.1 TPR-Tooth Program Register

TPR provides configuration for the Angle Counter circuit. In this register, the Engine CPU can properly adjust the tooth count (controlling last tooth, missing teeth, dummy tooth insertion, halt until tooth detection) and the number of angle ticks per tooth (field TICKS). Note that this register is sampled into a temporary register in the EAC logic when the High Rate Mode is detected (see High Rate Mode (Acceleration) and TPR Buffering), which means that changes to this register may take effect only for the next tooth.

Refer to Count Control and High Rate Logic and to Special Cases of Missing Teeth and Last Tooth until Handling False Tooth Detection for a detailed explanation about the use of this register. The figure below provides a detailed description of TPR register.

Several conflict issues on TPR writes are explained in Special TPR Write Cases.

**Figure 131. TBR_TPR register**




LAST —Last Tooth Indication

Asserted by Engine CPU and negated when a tooth is detected or inserted via IPH.

- 1 = Last Tooth - reset TCR2 Counter at the end of the tooth tick count (after physical tooth or IPH=1) when MISSCNT=0.
- 0 = Not Last Tooth.

If the tooth is detected or inserted before the tooth tick count completes (going High Rate mode, see High Rate Mode (Acceleration)), LAST resets immediately, but TCR2 resets only when the tooth count completes and MISSCNT=0.

MISSCNT — Missing Tooth Counter

Decremented on each estimated tooth, stops at zero. Used for generation of up to 7 "Dummy Teeth" whenever it holds a non-zero value.

If the tooth is detected or inserted before the missing tooth tick count completes (going High Rate mode, see High Rate Mode (Acceleration)), MISSCNT resets immediately, but missing teeth count continues in High Rate mode (see TPR Buffering).

IPH — Insert Physical Tooth

This bit generates a dummy physical tooth which has the same effect as a real physical tooth, and resets itself subsequently. If EAC is in Halt mode, it switches back to Normal mode1. If EAC is in Normal Mode, it switches to High Rate Mode. If Angle Logic is frozen by HOLD=1 (see below), it returns to the state it was at the freezing moment.

- 1 = Insert dummy physical tooth.
- 0 = No Operation.

***Note:*** IPH reads as 1 in the next microinstruction after it is asserted, negating subsequently. However, it can be set twice in two consecutive microinstructions to generate two teeth and make the EAC go from Halt to Normal to High Rate Mode.

HOLD — Force EAC Hold

This bit forces the EAC to halt its operation in a special EAC freeze mode until a new physical tooth (a real one or emulated with IPH=1) is detected. Assertion of this bit immediately freezes the EAC in the middle of the tooth period.

When a new physical tooth is detected, the bit is automatically negated by the EAC. The HOLD bit can be used for synchronizing the EAC tooth count, in case that a false physical tooth is detected due to noise.

- 1 = Force EAC to halt until detection of a physical tooth.
- 0 = Normal Operation.

TICKS — Angle Ticks Number in the Current Tooth

This field defines the number of angle ticks in the current physical tooth. It partitions the tooth period to the required number of angle ticks. The actual number of angle ticks in a tooth is (TICKS+1).

***Note:*** In High-Rate mode (see High Rate Mode (Acceleration)), TPR writes are immediately effective only for bits IPH and HOLD. All other fields changes are "buffered" and become effective when EAC leaves High Rate mode. See also Special TPR Write Cases.

***Note:*** Bits LAST, IPH and HOLD must not be asserted all at once.

1. missing a physical tooth naturally causes EAC to get into Halt mode.

#### 56.5.5.2.2 TCR2-Timer Counter Register

In Angle Mode, TCR2 (see the following figure) counts angle ticks instead of time.

**Figure 132. TBR_T2R register**

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TCR2 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TCR2 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

This 24-bit free-running counter is used to generate an accumulated Angle Fraction value. It is updated by the Angle Tick Generator - refer to Angle Tick Generator for more details. Refer to Count Control and High-Rate Logic for a detailed explanation about the use of this register in Angle Mode.

TCR2 provides continuous count of the angle in units of angle ticks. The Angle Tick Counter in TCR2 can be reset due to "Last Tooth" indication. TCR2 Prescaling is disregarded in Angle Mode: physical tooth detection is done by EAC regardless of the value set in TCR2P.

#### 56.5.5.2.3 TRR-Tick Rate Register

The exact period of the Angle Tick is programmed in the Tick Rate Register (see the figure below) by Engine CPU. The TRR time unit depends on the TBR_CR bit ATGC (see Figure 135. Angle counter system):

- if ATGC=0, the TRR time unit is the period of TCR1 clocks taken as TPU clocks divided by 2*(TCR1P + 1), even if TCR1CS=1 (see the TBR_CR register).
- if ATGC=1, the TRR time unit is given by the TPU clock divided by 2*(TCR2P+1), using the TCR2 prescaler fed by TPU clock/2 (independently of TCR1CS).

Refer to Calculating the Angle Tick Period Integer and Fraction for a complete description about the mechanism to calculate the value to be written into this register.

**Figure 133. TBR_TRR register**

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | INTEGER | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INTEGER | | | | | | | | FRACTION | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

INTEGER — The integer part of TCR1/TCR2 clocks in one Angle Tick.

This number, decremented by one, works as a down-counter preload value. A value of INTEGER=0 represents an integer of 32768. A new value written is reloaded into the counter (becoming effective) when a new tick starts or a tooth is detected or inserted via IPH.

FRACTION — Nine-bit fractional part of TCR1/TCR2 clocks in one angle tick.

The FRACTION value is accumulated in the EAC Fraction Accumulator, and whenever the result overflows (i.e., the accumulated fraction added up to an integer), the Tick Prescaler is halted for one TRR time unit.

**Figure 134. EAC PLL**

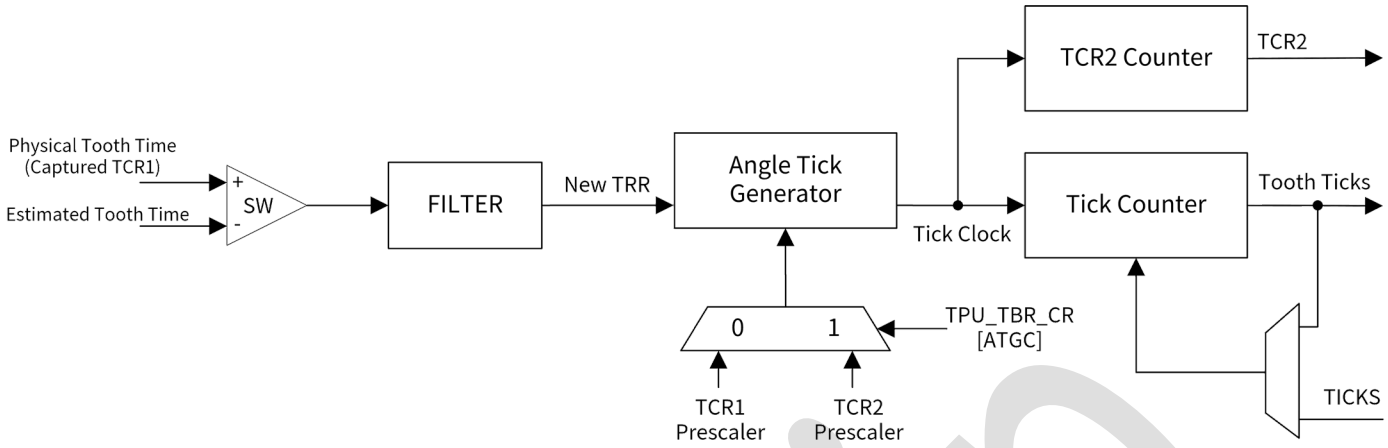
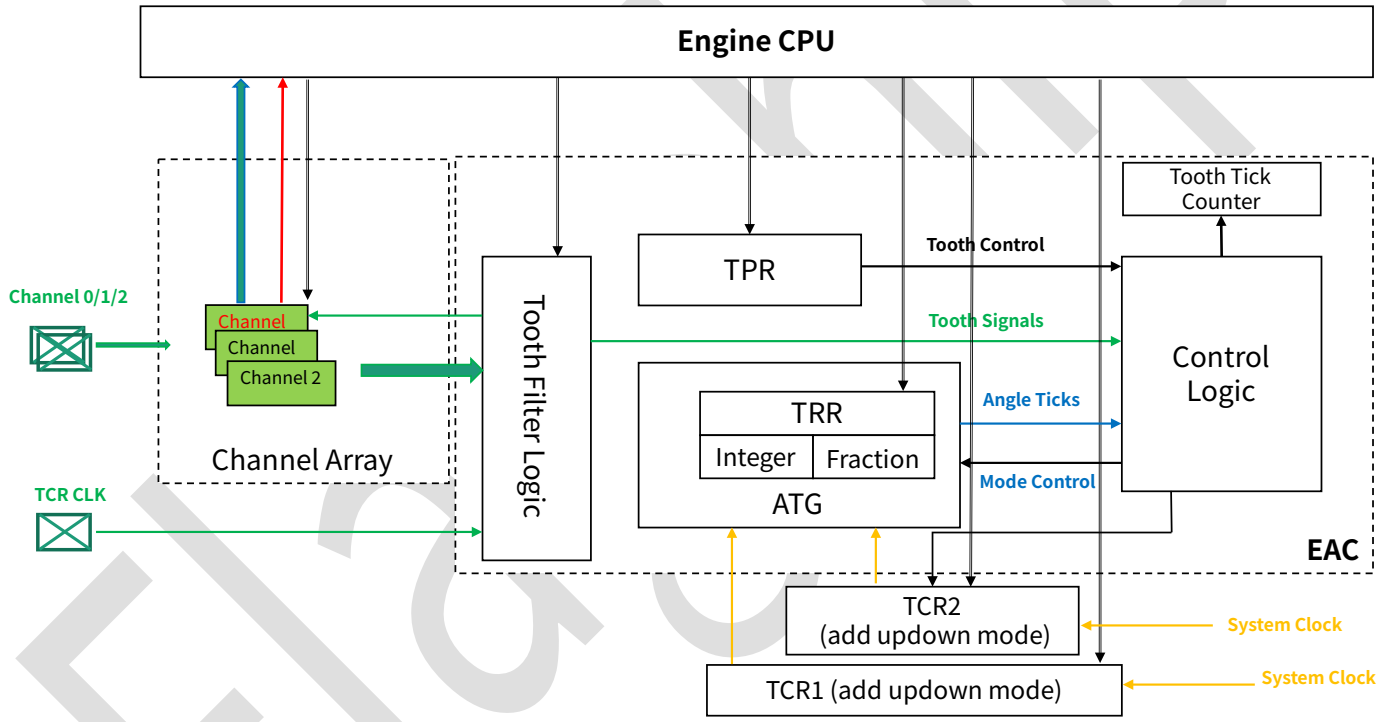



**Figure 135. Angle counter system**




### *56.5.5.3 Acceleration and Deceleration*

Acceleration and deceleration affect the new tooth period relative to the known period of the last tooth. Changes in tooth period may be extreme at very low engine RPM (such as cold start and warm start). The worst case of tooth period changes is caused during missing teeth, since there is more time for changes in angular velocity to be unnoticed by the EAC hardware. For example, on cold start (~20 RPM) there may be extreme acceleration: the ratio between a known tooth period before two missing teeth and the new tooth period after the missing teeth can be very high (up to a factor of 75). Acceleration and deceleration effects from tooth to tooth are less extreme as the engine climbs to high RPM.

In case of deceleration, the estimated tooth period ends before the actual tooth detection arrives. In this case, the EAC hardware waits at the end of the current tooth period, when it is said to be in Halt mode, until the real tooth indication is received, then continues with normal operation (Normal mode). See Figure 134. EAC PLL.

In case of acceleration, the actual tooth period is shorter than the estimated tooth period. As a result, a new physical tooth indication arrives before the end of the estimated tooth period. In this case the EAC closes the gap on High Rate mode by

counting on TPU clock divided by eight to the end of the tooth, advances to the next tooth, and switches back to normal operation mode. See Figure 134. EAC PLL.

The reason that the EAC does not jump directly to the next tooth is the need to provide sequential angle count throughout the whole tooth period, for channels clients which compare angle in "equal" mode. These peripherals must get all the valid angle values in a sequential manner, to avoid missing angle matches.

TCR2 advancing from one tooth to another is a continuous count, and can be optionally reset at the end of the tooth. An estimated tooth is generated after the Tooth Tick Counter reaches the TICKS programmed value.

The EAC works continuously and switches automatically between Normal, Halt and High-Rate modes. It relies on the Engine CPU to calculate the estimated tooth period on every tooth, and to update the correct angle tick and tooth parameters in the EAC control registers. On high RPM, tooth period changes are reduced from tooth to tooth, and the EAC may follow the angle with good accuracy for several teeth without microcode intervention.

The EAC handles missing teeth by insertion of "dummy" teeth, or by enlarging the expected tooth period. It is a good practice to locate the flywheel missing teeth in noncritical angles, since a missing tooth may increase the angle measurement error (acceleration and deceleration is detected late).

#### *56.5.5.4 Angle Tick Generator*

The Angle Tick Generator is responsible for generating a programmed number of angle ticks in the tooth period. It generates the ticks in an average rate which ensures completion of the correct number of angle counts in the estimated period of the tooth, since the count of one tooth in angle ticks is independent on engine RPM. The main
output of the Angle Tick Generator is the tick clock that feeds TCR2 in Angle Mode, as well as the internal Tooth Tick Counter (see Figure 13. Angle counter system). The Tooth Tick Counter counts ticks within a tooth, from 0 up to TICKS, is controlled by the Angle Tick Generator logic and cannot be accessed by microcode.

##### 56.5.5.4.1 Calculating the Angle Tick Period Integer and Fraction

On each tooth the microcode has to update the exact period of a single angle tick used for counting the internal angle in the tooth. The period of an angle tick is determined by the EAC configuration, see Generating the Angle Ticks for details.

The Engine CPU can use the MAC Divider unit to divide the tooth period by the number of angle ticks per tooth, which is stored in the TICKS field of TPR (refer to Angle Mode Registers). This division yields the integer part of the angle tick period and the remainder. Dividing again the remainder shifted left nine positions, by the number of angle ticks per tooth translates the remainder to a 9-bit fraction. The microcode concatenates the 15-bit integer and the 9-bit fraction to a 24-bit value and writes it to TRR. The new rate is effective immediately after the next Angle Tick is generated by the Angle Tick Generator.

For high RPM, note that shifting the tooth period value nine positions to the left prior to the first divide operation would calculate, in one operation, the integer and the fraction. For example: On 60 teeth flywheel running at 1000 RPM, tooth period is 1 msec. If TCR1/TCR2 counts @25 MHz, it counts 25,000 times in a tooth, which can be represented by 15 bits. Therefore the tooth period can be shifted nine positions to the left prior to divide operation, and be represented with 24 bits.

Using shift left nine positions and one divide operation would get the result which holds the integer and nine bits of the fraction:

Angle_Tick_Rate {Integer[14:0], Fraction[8:0]} = (ToothPeriod28 <<9) / Ticks

TRR = Angle_Tick_Rate {Integer[14:0], Fraction[8:0]}

On low RPM the initial tooth period, measured in TCR1/TCR2 counts, may be too big to be shifted nine positions to the left. For lower RPM (for example 500 RPM) the tooth period cannot be represented by 15 bits, and shifting it nine positions to the left would lose the MSB.

Angle_Tick_Rate {Integer[14:0]} = (ToothPeriod) / Ticks

Remainder[9:0] = ToothPeriod) mod Ticks

Angle_Tick_Rate {Fraction[8:0]} = (Remainder[9:0] << 9) / Ticks

TRR = Angle_Tick_Rate {Integer[14:0], Fraction[8:0]} = (Integer[14:0] << 9) | Fraction[8:0]

#### 56.5.5.4.2 Generating the Angle Ticks

The integer part of TRR is preloaded to a prescaler, which counts in time intervals as follows (see Figure 135. Angle counter system):

- if TBR_CR bit ATGC=0, one unit of TCR1 clock period (if TCR1CS=0) or twice the TCR1 clock period (if TCR1CS=1)29, that is, TPU clock period*2*(TCR1P+1) (independently of TCR1CS).
- if ATGC=1, two units of the TPU clock period multiplied by the TCR2 prescaler factor, that is, TPU clock period*2*(TCR2P+1).

When the down counter reaches zero, it generates an angle tick pulse to the Angle Counter Logic and a Load pulse to the Fraction Accumulator. It is then preloaded with most updated TRR integer part. Due to the Load pulse, the 9-bit fraction is accumulated in a 9-bit Fraction Accumulator. If a fraction overflow condition occurs (the 9-bit adder asserts carry out), the accumulator saves the lower 9 bits of the addition result, which is the remaining fractional part. The carry out bit indicates an accumulated integer "one" which means that the angle tick is early by one input clock. It halts the prescaler operation for one input clock to compensate the accumulated error generated by the integer prescaler. As a result, the average angle tick period takes into account both the integer and the fraction parts. The accuracy depends on the bit count of the fraction. Using 9-bit fraction part while the width of the field TICKS in register TPR is 10 bits provides accuracy of two LSB on a full scale (TICKS=1023) or one LSB on lower scale (TICKS<=511).

When the Tick Prescaler gets High Rate mode indication from the Angle Counter Logic, it generates angle ticks at a rate of TPU clock divided by eight. In this case it does not generate Load pulses to the Fraction Accumulator, ignores its "hold" input and preloads internally to a fixed period of eight TPU clocks. When High Rate mode is entered, the prescaler is preloaded to a period of eight TPU clocks before its first angle tick generation, ensuring separation of at least eight TPU clocks between the last Normal mode angle tick and the first High Rate mode angle tick. The fraction accumulator resets when the tick count advances to the next tooth, or when TRR is written by the Engine CPU.

### *56.5.5.5 Count Control and High-Rate Logic*

The Count Control and High Rate Logic controls TCR2 operation in Angle Mode, using the angle ticks generated by the Angle Tick Generator. Count Control logic is responsible for advancing, holding and resetting the TCR2 and Tooth Tick Counter in the proper timing, such that the TCR2 time base will reflect the correct estimated angle. This logic also includes the Tooth Program Register (TPR, see Angle Mode Registers for more information)

The Count Control and High Rate Logic handles deceleration, acceleration, missing teeth and last tooth. On High Rate (acceleration) it ensures that the angle bus scans all valid angle values in a rate which can be traced by the STAC bus. Because the TPU channels are capable of capturing either TCR1 or TCR2 due to signal transition, the microcode can get either the angle or time of the physical pin transition. Since the EAC channel (0, 1 or 2) is connected to the physical tooth, the microcode can get the EAC error in angle domain (tooth appears at the wrong angle) or time domain (physical tooth captured time into the EAC channel, relative to the estimated tooth time). Note that in angle mode, the transition detect logic

of the channel 0, if selected as the EAC channel, is fed from the digital filter of the TCRCLK signal, and not from the channel 0 internal digital filter. This ensures synchronous operation of channel 0 and the EAC hardware.

Another feature of the TPU channel, when working in Single Match And Single Transition Enhanced Mode (refer to Single Match Enhanced Mode (sm_st_e)), is capturing a single time base due to signal transition before and after the digital filter. This option allows subtracting the digital filter delay to get accurate signal transition timing on the channel. This way, the TCRCLK signal may be programmed with a slow and reliable digital filter, and get accurate time measurement of the digital filter delay.

To assert the end of the estimated tooth period the Count Control and High Rate logic compares the TICKS field in TPR (refer to Angle Mode Registers) with the current value of the tooth tick counter. When the Tooth Tick value is greater or equal to TICKS, it etermines the end of the estimated tooth period. On acceleration this event occurs during High-Rate mode operation, after the arrival of a physical tooth. In deceleration, this event occurs during Normal Mode, before the arrival of a physical tooth. On constant angular velocity, this event appears together with the arrival of a physical tooth.

The following sections describe the operation of the Counter Control and High Rate logic.

#### 56.5.5.5.1 Normal Mode

In Normal mode the Counter Control logic advances TCR2 and the Tooth Tick counter as if the engine has a constant speed during the tooth period. It receives the angle ticks from the Angle Tick Generator in an average rate which is determined by the Tooth Rate Register (TRR). This is the reset mode.

When the Tooth Tick Counter is about to reach the last value effectively stored in TPR field TICKS plus one, the hardware detects the end of the estimated tooth period. If the physical tooth and the estimated tooth arrive at the same time the EAC stays in Normal mode, the Tooth Tick Counter is reset, and TCR2 is incremented (depending on TPR bits LAST and MISSCNT). If the physical tooth and the estimated tooth do not arrive at the same time, either acceleration or deceleration is detected, and the EAC switches to the proper mode. See the figure below for a detailed diagram of Normal Mode behavior.

The Engine CPU which services the EAC channel physical tooth transition may update TRR according to various conditions to give the best estimation of the current tooth period, according to the previous tooth period and other engine parameters.

#### 56.5.5.5.2 Halt Mode(Deceleration)

In case of deceleration, the Tooth Tick Counter reaches the TICKS value before the arrival of the next tooth. The Count Control logic does not reset neither advances TCR2 and Tooth Tick Counters. The Count Control logic halts TCR2 and Tooth Tick Counters at the end of the tooth, waiting for the physical tooth to arrive.

When the physical tooth is detected the EAC switches back to Normal Mode and releases TCR2 to count the angle ticks of the new tooth, also resetting the Tooth Tick Counter. Only then TCR2 may wrap to 0, if TPR bit LAST is asserted. See the figure below for a detailed diagram of Halt Mode behavior.

The Engine CPU service caused by the physical tooth determines the deceleration, calculates the new tooth period and Angle Tick period and updates TRR. This operation slows the angle tick rate generated by the Angle Tick Generator on-the-fly, to the rate required for the new tooth period.

Since the Engine CPU service is initiated by the physical tooth edge, microcode latency may introduce a small angle error caused by using the TRR value of the previous tooth at the beginning of the current tooth. On high RPM, deceleration is relatively small but the Engine CPU latency may take a significant percentage of the tooth period. On low RPM Engine CPU service latency takes little percentage of the tooth period, but there may be cases of extreme acceleration and deceleration. The Engine CPU latency can be calculated knowing TCR1 value during the service time, and TCR1 value captured in the EAC channel due to the physical tooth pin transition. The duration of the Halt mode is obtained using the estimated tooth time.

#### 56.5.5.5.3 High Rate Mode (Acceleration)

In case of acceleration, the next tooth arrives before the Tooth Tick Counter reaches the TICKS value. In this case the High-Rate logic is responsible for closing the gap and advancing the tooth on the correct timing. The High-Rate mode operates as follows:

- When the acceleration is detected (physical tooth arrives before the Tooth Tick Counter reaches the TICKS value), the Count Control and High-Rate logic switches to High Rate mode in which both the Tooth Tick Counter and TCR2 count at rate of TPU clock divided by eight, until the Tooth Tick Counter reaches the current TICKS value. To ensure correct operation, the TICKS value is sampled in the logic at the beginning of the mode.
- At this point, which represents the estimated tooth edge, the logic resets the Tooth Tick Counter and advances TCR2 (or resets it if LAST is asserted and MISSCNT = 0).
- The control logic switches back to Normal Mode, using the most updated TRR value as input to the Angle Tick Generator. The logic samples the updated TICKS value for the tooth estimation, last tooth indication and number of missing teeth from TPR.

In High-Rate mode the angle ticks are provided at high speed until the end of the current tooth. This operation is required to scan all the valid angle values of the current tooth, in a rate which is not too high continuous update, but much higher than the rate dictated by TRR.

EAC channel microcode, which services the physical tooth transition detection, can start its service either before High Rate mode operation is complete (the Tooth Tick Counter has not reached the TICKS value) or after the EAC switched back to Normal mode. Any physical teeth received while the EAC is in High Rate Mode does not alter the immediate EAC state, but it is still detected by the EAC channel logic and can, therefore, alter future EAC behavior (for instance, closing the tooth detection window - see Angle Logic and Channel Modes).

At the beginning of High Rate mode operation, the TPR value is preloaded into a temporary register in the Counter Control Logic, used for scanning all the valid values to the end of the current tooth, with its appropriate LAST and MISSCNT attributes. While the EAC is in High Rate mode operation, the effect of microcode update of TPR fields LAST, MISSCNT and TICKS is delayed to the next estimated tooth, after the High Rate mode operation is complete (see Special TPR Write Cases). This is because the current physical tooth represents the next estimated tooth. If the Engine CPU updates this field after High Rate mode operation is complete, the current physical tooth and estimated tooth are the same, and the effect is immediate. Either in High-Rate mode or not, the value read by microengine is the same written, even if not yet effective, until the EAC resets LAST and/or IPH, or decrements MISSCNT. Typically the Engine CPU service may occur during the High Rate mode on extreme acceleration situation at low RPM.

Therefore, the Engine CPU operations are always related to the real physical tooth. From the above it can be seen that the Engine CPU updates of the TICKS field in TPR affect the end time of the current physical tooth. For correct operation, this field should be updated before the Tooth Tick Counter has reached either the old or the new TICKS value.

During High-Rate mode operation, TRR is ignored and the Angle Tick Generator uses TPU clock divided by eight. Therefore, the TRR update by microcode will take effect only after the EAC switches back to Normal mode. If Engine CPU service occurs after the Tooth Tick Counter has been reset, the EAC is already back in Normal mode, and some angle ticks may have been counted at the rate of the previous tooth. In this case the new TRR value will have immediate effect on the angle tick period, and the microcode should take into consideration the delay from the physical tooth to the estimated tooth in calculation of the next tooth period. See the figure below for a detailed diagram of High-Rate Mode behavior.

An angle error may be introduced by the duration of the High-Rate mode. Also, the scheduler latency may introduce a small accumulated error by using TRR value of the previous estimated tooth at the beginning of the current tooth. After the

estimated tooth has advanced, the duration of the High-Rate mode operation is the actual delay from the physical tooth edge to the estimated tooth edge. This delay can be obtained by comparing the estimated tooth time with the EAC channel capture register which captured TCR1 on the physical pin transition.

#### *56.5.5.6 Special Cases of Missing Teeth and Last Tooth*

The EAC handles cases of up to three missing teeth and the last tooth in the engine cycle. The following paragraphs describe these functions.

##### 56.5.5.6.1 Handling the Last Tooth

The Engine CPU can set the TCR2 counter to work in engine periods (wrap-around count) or continuous angle measurement.

For periodic operation, during the last engine cycle tooth the EAC microcode has to set the LAST flag in TPR. As a result, when the tooth period is ended, the Counter Control Logic generates a reset command to both the Tooth Tick Counter and TCR2 instead of an advance command. The operation resets the TCR2 based angle count, indicating a new period of the engine cycle. This implementation provides an engine cycle based periodic angle measurement.

##### 56.5.5.6.2 Handling Missing Teeth

The EAC can handle up to three missing teeth in two ways:

- Count the angle ticks relative to the last physical tooth. The Engine CPU should update the TPR TICKS field to the number of angle ticks included in two, three or four teeth, according to the flywheel type (one, two or three missing teeth). EAC hardware works in its regular manner.
- Insert a "dummy" tooth instead of the missing tooth, at the estimated point in time. After the "dummy" tooth, the Angle Tick Counter is incremented as if there was a physical tooth. A "dummy" tooth can be inserted only during Normal or High-Rate operation modes. The microcode inserts "dummy" teeth by writing to the MISSCNT field in TPR. In the first option the missing tooth is not counted on the angle measurement. For example, a flywheel with 59 physical teeth and one missing tooth can be considered as 58 identical teeth numbered (0-57) and tooth number 58 has a double number of angle TICKS. In this case a 720 degrees engine cycle has 118 teeth. TCR2 reflects the real angle, since it counts angle ticks continuously.

In the second option, the missing teeth are counted as "regular" teeth by automatic insertion of "dummy" teeth. The Engine CPU has to write a non-zero value to the MISSCNT field in TPR. This field is a 3-bit down counter which affects the operation of the Counter Control logic. For example, a flywheel with 59 physical teeth (0-58) and one missing tooth (59) can be considered as 60 teeth numbered (0-59), all having the same number of angle ticks. The Engine CPU has to write "001" to the MISSCNT bits during the period of tooth number 58 to indicate that next tooth (59) is missing.

When the Tooth Tick Counter reaches the TICKS value, TCR2 is incremented as if a physical tooth has been detected. In addition, the MISSCNT value initializes a "dummy tooth counter" which is decremented to indicate the number of left "dummy teeth" which still need to be generated. Because a dummy tooth was counted, EAC does not enter Halt Mode and Tooth Tick Counter continues incrementing in the absence of a physical tooth detection.

In case of extreme acceleration on very low RPM (cold start) there can be a situation that the first physical tooth after one or two missing teeth appears even before the "dummy" tooth is generated. Due to the acceleration the EAC switches to High-Rate mode in order to run through all the valid angle values, including the dummy teeth. When the Tooth Tick Counter reaches the TICKS value on High-Rate mode, and the "dummy tooth" down counter is not zero, the generated "dummy tooth" advances to the next tooth and decrements the "dummy tooth" counter, but does not switch the EAC back to Normal mode. The last "dummy tooth" decrements the counter to zero, indicating that no more dummy teeth are to be inserted, and the next tooth is an estimated physical tooth. The EAC continues at High-Rate mode until the Tooth Tick Counter reaches the

TICKS value again, then advances to the next tooth while switching back to Normal mode. When in High-Rate mode, the TPR register does not reflect the MISSCNT downcounting, see TPR Buffering for details.

MISSCNT can be rewritten before it reaches 0, allowing it to count more than 7 missing teeth, as long as no physical tooth arrives between the first MISSCNT write and the rewrite.

#### 56.5.5.6.3 Combining Missing Teeth and Last Tooth

The Last Tooth indication takes effect when there are no more missing teeth to be generated, i.e the "dummy tooth" counter value is zero. If, for example, the Engine CPU sets the missing teeth counter to "10" (two missing teeth) and sets the LAST flag, the first and the second dummy teeth will increment TCR2, and the third estimated tooth, which correlates with the physical tooth (the first of the next cycle), will reset TCR2, because LAST was set. This scheme enables the microcode to define one or more missing teeth to be replaced by "dummy tooth" insertion, and the end of the engine cycle in one service request. It is assumed that the two missing teeth must come together in the same engine cycle, and not split between two engine cycles (either the missing teeth are both last in an engine cycle or both not last, but not last in one engine cycle and first in the next).

### *56.5.5.7 Handling Mechanical Tooth Correction*

The EAC can handle tooth edge detection errors caused by flywheel mechanical errors. The TPU application can hold a vector of tooth mechanical errors with one entry per tooth. This error can be measured in angle ticks which are independent of engine RPM. The TRR can be updated to the fixed period of any tooth, including its mechanical error.

Because TCR2 counts continuously, without being reset, the mechanical correction is transparent. Though the tooth has its own programmed TICKS value, TCR2 simply counts angle ticks, disregarding the boundary between two adjacent teeth.

### *56.5.5.8 Handling Mis-detected Tooth*

When a physical tooth signal is missed by the engine sensor, the EAC may get into Halt mode at the end of the estimated tooth period, expecting the physical arrival. In this case, a Match timeout event of EAC channel will call service which detects extreme deceleration. The microcode can assert the IPH bit in TPR, to force the detection of the missed physical tooth. It can also calculate the accumulated angle bus error, and fix the next estimated tooth period, to close the gap.

### *56.5.5.9 Handling False Tooth Detection*

Most of the false tooth detection, caused by noises on the engine tooth sensor, can be eliminated by the window blanking filtering, timed by EAC channel match recognitions. The EAC also provides means of fixing false detection of an additional tooth which passed the window filter. When such an event occurs, the EAC switched to High Rate mode (advancing to the next tooth) and when the next physical tooth arrives, an extreme acceleration is detected: the EAC sees the remaining portion of the current tooth period as another tooth period. The microcode can detect the situation when the acceleration in not realistic, or when immediately after the detection of this extreme acceleration, the following tooth indicates extreme deceleration back to the original RPM.

When the microcode detects such a case, the Tooth Counter has been advanced by mistake to the next tooth. The Engine CPU can set the HOLD bit in the TPR, forcing the EAC to freeze and wait for the next physical tooth to close the gap. When the next physical tooth arrives, HOLD is automatically negated and the EAC proceeds from that point to the remaining portion of the tooth period, in the same mode it was when HOLD bit was asserted.

### *56.5.5.10 Angle Logic and Channel Modes*

The TCRCLK transition detection is qualified by a signal that comes from channel 0 (see Figure 135. Angle counter system) and depends on the particular mode (PDCM) programmed for that channel. This configures a window for TCRCLK detection for the angle logic which is the same (except on High-Rate mode, see High-Rate Mode (Acceleration)) used to set the TDL1

flag on single transition modes, and TDL2 on double transition modes. The same applies when channels 1 or 2 are used to control EAC. As a consequence, the window depends on the channel mode as follows:

- on all modes, the window closes upon a tooth edge detection: TDL1 asserted on single transition modes, TDL2 asserted on double transition modes.
- on mode m2_st: the window opens on Match 1 (which enables Transition 1) and does not close with Match 2. If Match 2 comes before Match 1, it blocks Match 1 and, hence, Transition 1.
- on mode m2_o_st: the window opens on Match 1 (which enables Transition 1) and closes on Match 2. Match 2 is enabled by Match 1, so it cannot come before.
- on all other Single Transition modes, the window is "always open", independently of matches.
- on mode m2_dt: the window opens on Transition 1, which is enabled by Match 1. The window does not close with Match B, but if it comes before Match A the later gets blocked and, hence, blocks Transitions. Match 1 is also a condition for the window, so the Engine CPU closes it by clearing MRL1.
- on mode m2_o_dt: the window opens on Transition 1, which is enabled by Match 1. The window closes on Match 2, which is enabled by Match 1. Match 1 is also a condition for the window, so the microcode closes it by clearing MRL1.
- on all other Double Transition modes, the window opens on Transition 1.

#### *56.5.5.11 Restarting Angle Logic*

It is not recommended to change the ETPUTBCR field AM while GTBE=1. However, if the Angle Logic must be restarted without interfering with the timebase count running on TCR1, the procedure below must be followed:

1) write TBR_CR setting AM=00 and TCR2CTL=111 at once. That prevents TCR2 from incrementing while the Angle logic is disabled. The Angle Logic state-machine resets to Normal mode and the tick prescaler to the initial count with AM=00.
2) start a thread to reconfigure the EAC. The thread must set the EAC controlling channel (0, 1 or 2) flags in a state, depending on the channel mode, that lets the channel tooth detection window open (see Angle Logic and Channel Modes). It can optionally write TCR2 with an angle preset value equivalent to the first tooth expected after restart. The thread must also set TPR bit HOLD=1. The TPR bit IPH must be 0.
3) after the thread is finished, write ETPUTBCR setting AM=01, 10 or 11, and TCR2CTL according to the desired tooth edge selection if AM=01.

The first tooth detected after this procedure restarts the TCR2 counting, unfreezing the Angle Mode logic into normal mode.

#### *56.5.5.12 Special TPR Write Cases*

This section describes how simultaneous modification of TPR fields are resolved, and how the effect of TPR writes depend on the EAC mode.

##### 56.5.5.12.1 TPR Buffering

In High-Rate mode (see High-Rate Mode (Acceleration)), TPR writes are immediately effective only for bits IPH and HOLD. Writes to all other fields are "buffered" and become effective when EAC leaves High-Rate mode. However, if TPR is written a second time right after IPH is asserted in Normal mode, this second write behaves as if EAC is still in Normal mode.

MISSCNT and LAST can be written any value during High-Rate mode, and the value that prevails for the next tooth is the one sampled when EAC goes back to Normal mode (or the value written in Normal or Halt mode thereafter). If MISSCNT and/or LAST are not zero when High-Rate mode begins, they are sampled into the internal EAC logic and are effective while High-Rate lasts (missing teeth count continues, and TCR2 is reset at the end of High Rate if LAST=1). However, their values in TPR are reset when High-Rate mode starts. After that and until the end of High-Rate mode, the value read by Engine CPU is the same written. This behavior prevents read-modify-writes to TPR from unwillingly rewriting LAST or MISSCNT.

#### 56.5.5.12.2 IPH and Last

If both IPH and LAST are asserted in the same instruction with HOLD=0, the EAC acts as if LAST was set first and then IPH right after, so that:

- in Normal mode, it goes to High Rate with LAST=1;
- in Halt mode, it goes to Normal Mode resetting LAST (and TCR2);
- in High Rate mode, IPH is ignored and LAST becomes effective in the next tooth (physical or inserted) after it goes back to Normal mode.

#### 56.5.5.12.3 IPH and TICKS

Because of different results depending on the EAC mode at the time of TPR write, it is not advisable to write 1 to IPH and change TICKS at the same microinstruction. A consistent behavior is obtained if IPH is written first and TICKS on the second instruction after (for instance with a nop between them), making the new TICKS value valid for the next tooth regardless of the mode. The mode-dependent behavior is (assuming HOLD=0):

- in Normal mode, the new TICKS value becomes valid before EAC goes High Rate due to the IPH;
- in Halt mode, the EAC goes to Normal mode, and new TICKS is valid for the next tooth;
- in High Rate mode, the new TICKS value is effective when EAC leaves High Rate mode, and IPH is ignored.

#### 56.5.5.12.4 IPH and MISSCNT

If both IPH and MISSCNT are written non-zero values with HOLD=0:

- in Normal mode, at the next instruction cycle the EAC goes to High Rate mode, the MISSCNT field in TPR goes to 0, and the missing teeth are counted in High Rate mode.
- in Halt mode, the EAC goes to Normal mode for one instruction cycle and then, yet another instruction cycle later, goes to High Rate mode, counting the missing teeth. The TPR fields IPH and MISSCNT are zeroed on the transition from Normal to High-Rate mode.
- in High-Rate mode, IPH is ignored (resetting at the next instruction cycle) and MISSCNT is buffered (see TPR Buffering).

#### 56.5.5.12.5 IPH and HOLD

If IPH and HOLD are asserted at once, IPH cancels the HOLD and both reset. The EAC is not frozen, regardless of the EAC mode (High Rate, Normal or Halt).

#### 56.5.5.12.6 LAST and HOLD

If LAST and HOLD are written 1 at once, LAST asserts and EAC is frozen. Afterwards, when a physical tooth is detected or IPH is asserted, the EAC is unfrozen in the same state it was before, and LAST is kept asserted.

## 56.6 TPU_H Register Definition

TPU_H base address : 4044_5000h

**Register Overview - TPU_H**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **10h** | TPU_H_GCR_VSR (Global Control Region -Visible Status Register) | 32 | RO | 0000_0000h | page |
| **14h** | TPU_H_GCR_VIR (Global Control Region - Visible Input Status Register) | 32 | RO | 0000_0000h | page |

| | | | | | |
|---|---|---|---|---|---|
| **18h** | TPU_H_GCR_VOR (Global Control Region - Visible Output Status Register) | 32 | RO | 0000_0000h | page |
| **1Ch** | TPU_H_GCR_VOBR (Global Control Region - Visible Output Enable Status Register) | 32 | RO | 0000_0000h | page |
| **20h** | TPU_H_GCR_VM1R (Global Control Region - Visible MRL1 Register) | 32 | RO | 0000_0000h | page |
| **24h** | TPU_H_GCR_VM2R (Global Control Region -Visible MRL2 Register) | 32 | RO | 0000_0000h | page |
| **28h** | TPU_H_GCR_VT1R (Global Control Region - Visible TDL1 Register) | 32 | RO | 0000_0000h | page |
| **2Ch** | TPU_H_GCR_VT2R (Global Control Region - Visible TDL2 Register) | 32 | RO | 0000_0000h | page |
| **30h** | TPU_H_GCR_EM1R (Global Control Region - Visible Event by MRL1 Register) | 32 | RO | 0000_0000h | page |
| **34h** | TPU_H_GCR_EM2R (Global Control Region - Visible Event by MRL2 Register) | 32 | RO | 0000_0000h | page |
| **38h** | TPU_H_GCR_ET1R (Global Control Region - Visible Event by TDL1 Register) | 32 | RO | 0000_0000h | page |
| **3Ch** | TPU_H_GCR_ET2R (Global Control Region - Visible Event by TDL2 Register) | 32 | RO | 0000_0000h | page |
| **40h** | TPU_H_GCR_VHSR (Global Control Region - Visible Host Acknowledge Register) | 32 | RO | 0000_0000h | page |
| **100h** | TPU_H_TBR_CR (Time Bases Region - Control Register) | 32 | RO | 0000_0000h | page |
| **104h** | TPU_H_TBR_T1R (Time Base Region - TCR1 Value Register) | 32 | RW | 0000_0000h | page |
| **108h** | TPU_H_TBR_T2R (Time Base Region - TCR2 Value Register) | 32 | RW | 0000_0000h | page |
| **110h** | TPU_H_TBR_TPR (Time Base Region - Tooth Program Register) | 32 | RO | 0000_0000h | page |
| **118h** | TPU_H_TBR_TRR (Time Base Region - Tick Rate Register) | 32 | RW | 0000_0000h | page |
| **11Ch** | TPU_H_TBR_T1MR (Time Base Region - TCR1 Maximum Register) | 32 | RW | 00FF_FFFFh | page |
| **114h** | TPU_H_TBR_T2MR (Time Base Region - TCR2 Maximum Register) | 32 | RW | 00FF_FFFFh | page |
| **800h+n*40h** | TPU_H_CHn_CR (Channel N Control Register) | 32 | RW | 0000_0000h | page |
| **804h+n*40h** | TPU_H_CHn_SR (Channel N Status Register) | 32 | RO | 0000_0000h | page |
| **808h+n*40h** | TPU_H_CHn_SCR (Channel N Status Control Register) | 32 | W1C | 0000_0000h | page |
| **80Ch+n*40h** | TPU_H_CHn_EFR (Channel N Event and Flag Register) | 32 | RO | 0000_0000h | page |

### 56.6.1 TPU_H_GCR_VSR (Global Control Region -Visible Status Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SR | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_GCR_VSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**SR** | RO | Each bit indicates status of a channel.<br>For example, bit 31 indicates the channel 31's service request status |

### 56.6.2 TPU_H_GCR_VIR (Global Control Region - Visible Input Status Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PSTI | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PSTI | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_GCR_VIR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**PSTI** | RO | Each bit indicates status of a channel.<br>For example, bit 31 indicates the channel 31's pin input status. |

### 56.6.3 TPU_H_GCR_VOR (Global Control Region - Visible Output Status Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PSTO | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PSTO | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_GCR_VOR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**PSTO** | RO | Each bit indicates status of a channel.<br>For example, bit 31 indicates the channel 31's pin output status. |

### 56.6.4 TPU_H_GCR_VOBR (Global Control Region - Visible Output Enable Status Register)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PSTOE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PSTOE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_GCR_VOBR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**PSTOE** | RO | Each bit indicates status of a channel.<br>For example, bit 31 indicates the channel 31's pin output enable status. |

### 56.6.5 TPU_H_GCR_VM1R (Global Control Region - Visible MRL1 Register)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | M1R | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | M1R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_GCR_VM1R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**M1R** | RO | Each bit indicates status of a channel.<br>For example, bit 31 indicates the channel 31's MRL1. |

### 56.6.6 TPU_H_GCR_VM2R (Global Control Region -Visible MRL2 Register)

Offset : 24h

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| R | M2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | M2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_GCR_VM2R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**M2R** | RO | Each bit indicates status of a channel.<br>For example, bit 31 indicates the channel 31's MRL2. |

### 56.6.7 TPU_H_GCR_VT1R (Global Control Region - Visible TDL1 Register)

Offset : 28h

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| R | T1R | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | T1R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_GCR_VT1R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**T1R** | RO | Each bit indicates status of a channel.<br>For example, bit 31 indicates the channel 31's TDL1. |

### 56.6.8 TPU_H_GCR_VT2R (Global Control Region - Visible TDL2 Register)

Offset : 2Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | T2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | T2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_GCR_VT2R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**T2R** | RO | Each bit indicates status of a channel.<br>For example, bit 31 indicates the channel 31's TDL2. |

### 56.6.9 TPU_H_GCR_EM1R (Global Control Region - Visible Event by MRL1 Register)

Offset : 30h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EM1R | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | EM1R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_GCR_EM1R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**EM1R** | RO | Each bit indicates status of a channel.<br>For example, bit 31 indicates the channel 31's event by MRL1. |

### 56.6.10 TPU_H_GCR_EM2R (Global Control Region - Visible Event by MRL2 Register)

Offset : 34h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EM2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | EM2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_GCR_EM2R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**EM2R** | RO | Each bit indicates status of a channel.<br>For example, bit 31 indicates the channel 31's event by MRL2. |

### 56.6.11 TPU_H_GCR_ET1R (Global Control Region - Visible Event by TDL1 Register)

Offset : 38h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ET1R | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ET1R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_GCR_ET1R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**ET1R** | RO | Each bit indicates status of a channel.<br>For example, bit 31 indicates the channel 31's event by TDL1. |

### 56.6.12 TPU_H_GCR_ET2R (Global Control Region - Visible Event by TDL2 Register)

Offset : 3Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ET2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ET2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_GCR_ET2R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**ET2R** | RO | Each bit indicates status of a channel.<br>For example, bit 31 indicates the channel 31's event by TDL2. |

### 56.6.13 TPU_H_GCR_VHSR (Global Control Region - Visible Host Acknowledge Register)

Offset : 40h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | HAS | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | HAS | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_GCR_VHSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**HAS** | RO | Each bit indicates status of a channel.<br>For example, bit 31 indicates the channel 31's HAS. |

### 56.6.14 TPU_H_TBR_CR (Time Bases Region - Control Register)

Offset : 100h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TCR1P | | | | | | | | 0 | | | | | TCR1CTL | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TCR2P | | | | | | 0 | | ATGC | AM | | TCRCF | | TCR2CTL | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_TBR_CR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**TCR1P** | RO | TCR1 Prescaler Control<br>TCR1 is clocked from the output of a prescaler. The prescaler divides its input by (TCR1P + 1) allowing frequency divisions from 2 to 256. The prescaler input is the TPU clock divided by 2, TPU clock or output of TCRCLK filtered. |

| | | |
|---|---|---|
| **18-16**<br>**TCR1CTL** | RO | TCR1 Clock/Gate Control<br>TCR1CTL = 3'b011 was updown counter mode<br>000b - Select TCRCLK as the clock source for the TCR1 prescaler<br>011b - TCR1 updown counter mode<br>100b - Select TPU clock divided by 2 as the clock source for the TCR1 prescaler<br>101b - Select TPU clock as clock source for the TCR1 prescaler |
| **15-10**<br>**TCR2P** | RO | TCR2 Prescaler Control<br>TCR2 is clocked from the output of a prescaler. The prescaler divides its input by (TCR2P + 1) allowing frequency divisions from 1 to 64. The prescaler input is the TPU clock divided by 8, or Internal Timebase input, or TCRCLK filtered input. No effect on Angle mode. |
| **7**<br>**ATGC** | RO | Angle Tick Generator Clock.<br>0b - TCR1 prescaler output provides time intervals for tick generation,<br>1b - TCR2 prescaler output provides time intervals for tick generation. |
| **6-5**<br>**AM** | RO | Angle Mode Selection<br>00b - TCR2 was timebase (EAC disabled). Tooth signal and tooth processing channel was not applicable.<br>01b - TCR2 was angle ticks, Tooth signal was TCRCLK input and tooth processing channel was 0.<br>10b - TCR2 was angle ticks, Tooth signal was Channel 1 input and tooth processing channel was 1.<br>11b - TCR2 was angle ticks, Tooth signal was Channel 2 input and tooth processing channel was 2. |
| **4-3**<br>**TCRCF** | RO | TCRCLK signal Filter Control<br>This field controls the TCRCLK digital filter, determining whether the TCRCLK signal input (after a synchronizer) is filtered with the same filter clock as the channel input signals or uses the TPU clock divided by 2.<br>00b - Two sample mode, Filter clock uses TPU clock divided by 2.<br>01b - Two sample mode, Filter clock uses the Filter clock of the channels.<br>10b - Integrator, Filter clock uses TPU clock divided by 2.<br>11b - Integrator, Filter clock uses the Filter clock of the channels. |

| | | |
|---|---|---|
| **2-0**<br>**TCR2CTL** | RO | TCR2 Clock/Gate Control<br>These bits are part of the TRC2 clock system. They determine the clock source for TCR2 before the prescaler. TCR2 can count on any detected edge of the TCRCLK signal or use it for gating TPU clock divided by 8. TCR2CTL also determines the TCRCLK edge selected for angle tooth detection in angle mode.<br>When AM=0, TCR2 clock before prescaler depends on TCR2CTL:<br>000b - Gated DIV 8 clock. In this case, when the external TCRCLK signal is low, TCR2 prescaler is incremented at the frequency of the TPU clock divided by 8.<br>001b - Rise transition on TCRCLK signal increments the TRC2 prescaler.<br>010b - Fail transition on TCRCLK signal increments the TRC2 prescaler.<br>100b - DIV8 clock<br>updown counter mode works on TCR2CTL = 3'b101<br>111b - TCR2 frozen, except as STAC client.<br>When AM=1, TCR2 was used for angle tooth detection.<br>001b - Rising edge<br>010b - Failling edge<br>011b - Both edges<br>110b - No edge<br>For other conditions, do not use. |

### 56.6.15 TPU_H_TBR_T1R (Time Base Region - TCR1 Value Register)

Offset : 104h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TCR1 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TCR1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_TBR_T1R field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-0**<br>**TCR1** | RO | TCR1 Counter Value |

### 56.6.16 TPU_H_TBR_T2R (Time Base Region - TCR2 Value Register)

Offset : 108h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TCR2 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TCR2 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_TBR_T2R field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-0 TCR2** | RO | TCR2 Counter Value |

### 56.6.17 TPU_H_TBR_TPR (Time Base Region - Tooth Program Register)

Offset : 110h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LAST | MISSCNT | | IPH | HOLD | TPR10 | TICKS | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_TBR_TPR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **15**<br>**LAST** | RO | Last Tooth Indication<br>0b - Not Last Tooth.<br>1b - Last Tooth: reset TCR2 on the next physical tooth edge (or when IPH=1) when MISSCNT =0. |
| **14-13**<br>**MISSCNT** | RO | Missing Tooth Counter<br>00b - Not a Missing Tooth.<br>01b - One Missing Tooth.<br>10b - Two Missing Tooth.<br>11b - Three Missing Tooth. |
| **12**<br>**IPH** | RO | Insert Physical Tooth<br>This bit is used for exiting halt mode which is caused by missing a detection of a physical tooth. It generates a dummy physical toothwhich has the same effect as a real physical tooth, and resets itself subsequently. If EAC is in halt mode, it switches back to normal mode. If EAC is in normal mode, it switches to high rate mode.<br>0b - No Operation.<br>1b - Insert dummy physical tooth. |
| **11**<br>**HOLD** | RO | Force EAC Halt<br>0b - Normal Operation.<br>1b - Force EAC to halt until detection of a physical tooth. |
| **10**<br>**TPR10** | RO | Reserved in Angle Mode. |
| **9-0**<br>**TICKS** | RO | Angle Ticks Number in the Current Tooth |

### 56.6.18 TPU_H_TBR_TRR (Time Base Region - Tick Rate Register)

Offset : 118h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | INTEGER | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INTEGER | | | | | | | | FRACTION | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_TBR_TRR field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-9**<br>**INTEGER** | RW | The integer part of TCR1 clocks in one Angle tick.<br>This number, decremented by one, is a down-counter preload value. A value of INTEGER=0 represents an integer of 32768. |
| **8-0**<br>**FRACTION** | RW | Nine-bit fractional part of TCR1 clocks in one angle tick.<br>The FRACTION value is accumulated in the EAC Fraction Accumulator, and whenever the result overflows. The tick prescaler is halted for one TCR1 clock. |

### 56.6.19 TPU_H_TBR_T1MR (Time Base Region - TCR1 Maximum Register)

Offset : 11Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OVF | 0 | | | IRQ_EN | 0 | | | MAX | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAX | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**TPU_H_TBR_T1MR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**OVF** | RO | TCR1 Overflow Flag<br>W1C.<br>0b - No overflow flag<br>1b - Overflow flag |
| **27**<br>**IRQ_EN** | RO | TCR1 Overflow IRQ Enable<br>0b - Overflow IRQ disable<br>1b - Overflow IRQ enable |
| **23-0**<br>**MAX** | RO | Maximum value of TCR1 counter in TCR1 updown mode |

### 56.6.20 TPU_H_TBR_T2MR (Time Base Region - TCR2 Maximum Register)

Offset : 114h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OVF | 0 | | | IRQ_EN | 0 | | | MAX | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAX | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**TPU_H_TBR_T2MR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**OVF** | RO | TCR2 Overflow Flag<br>W1C.<br>0b - No overflow flag<br>1b - Overflow flag |
| **27**<br>**IRQ_EN** | RO | TCR2 Overflow IRQ Enable<br>0b - Overflow IRQ disable<br>1b - Overflow IRQ enable |
| **23-0**<br>**MAX** | RO | Maximum value of TCR2 counter in TCR2 updown mode |

### 56.6.21 TPU_H_CHn_CR (Channel N Control Register)

Offset : 800h+N*40h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CHAE | CHDE | CHEE | CTC | | | | CDFD | 0 | | | | | CHSR | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_CHn_CR field descriptions**

| Field | Type | Description |
|---|---|---|
| **15**<br>**CHAE** | RW | Channel HSA Enable<br>0b - Disable<br>1b - Enable |
| **14**<br>**CHDE** | RW | Channel HSR DMA Request Enable<br>0b - Channel HSR DMA request disable<br>1b - Channel HSR DMA request enable |
| **13**<br>**CHEE** | RW | Channel HSR Event as Interrupt Enable<br>0b - Channel HSR interrupt request disable<br>1b - Channel HSR interrupt request enable |
| **12-9**<br>**CTC** | RW | Channel Trigger Configuration<br>0000b - Disable<br>0001b - Reserved<br>0010b - Any of Event happens, gated by MSR/TSR<br>0011b - Open on FlexCore mode, HSA happens<br>0100b - MRL1 event happens, not gated by MSR<br>0101b - MRL2 event happens, not gated by MSR<br>0110b - MRL1 or MRL2 event happens, not gated by MSR<br>0111b - MRL1 and MRL2 event both happens, not gated by MSR<br>1000b - TDL1 or TDL2 event happens, not gated by TSR<br>1001b - TDL1 and TDL2 event both happens, not gated by TSR<br>1010b - Euqal to channel do, level signal<br>1011b - Negative to channel do, level signal<br>1100b - Equal to channel do of part 1, level signal<br>1101b - Equal to channel do of part 2, level signal<br>1110b - Equal to channel ind, level signal<br>1111b - Equal to channel ind which latch until next event isr happens, level signal |
| **8**<br>**CDFD** | RW | Channel Event Triggered DMA Request Enable on FlexCore Disable Mode<br>0b - DMA not active by event<br>1b - DMA request active by event |
| **2-0**<br>**CHSR** | RW | Channel HSR index<br>0x00b - No HSR<br>Others - Has HSR |

### 56.6.22 TPU_H_CHn_SR (Channel N Status Register)

Offset : 804h+N*40h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CIS | 0 | | | | | | CHRS | 0 | | | | | CHRI | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | CEIS | CHDS | 0 | CTS | 0 | | | CHAS | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_CHn_SR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**CIS** | RO | Channel Event Trigger Interrupt Status<br>0b - no event trigger interrupt<br>1b - event trigger interrupt |
| **24**<br>**CHRS** | RO | Channel Host Service Request (HSR) Status<br>0b - No HSR<br>1b - HSR generated |
| **18-16**<br>**CHRI** | RO | Channel HSR Info & Index<br>Index was 18:16.<br>000b - No HSR<br>Others - Has HSR |
| **15**<br>**CEIS** | RO | Channel Event as Interrupt Status<br>0b - Channel has no interrupt request by event<br>1b - Channel has interrupt request by request |
| **14**<br>**CHDS** | RO | Channel Host Service Acknowledge (HSA) DMA Request Status<br>0b - Channel has no DMA request<br>1b - Channel has DMA request |
| **12**<br>**CTS** | RO | Channel Trigger Status<br>0b - Channel has no trigger request<br>1b - Channel has trigger request |
| **8**<br>**CHAS** | RO | Channel Host Service Acknowledge (HSA) Interrupt Status<br>0b - Channel has no HSA interrupt request<br>1b - Channel has HSA interrupt request |

### 56.6.23 TPU_H_CHn_SCR (Channel N Status Control Register)

Offset : 808h+N*40h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CEIC | 0 | | | | | | CHAC | 0 | | | | | | | CHRT |
| W | w1c | | | | | | | w1c | | | | | | | | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_CHn_SCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **15**<br>**CEIC** | W1C | Channel Event as Interrupt Clear<br>Write1 to clear channel interrupt by event. |
| **8**<br>**CHAC** | W1C | Channel HSA Clear<br>Write 1 to clear channel HSA. |
| **0**<br>**CHRT** | W1C | Channel HSR Request Trigger<br>Write 1 to clear HSR request. |

### 56.6.24 TPU_H_CHn_EFR (Channel N Event and Flag Register)

Offset : 80Ch+N*40h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CSR | CHSR | 0 | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | EMRL1 | EMRL2 | ETDL1 | ETDL2 | MRLE1 | MRLE2 | MRL1 | MRL2 | TDL1 | TDL2 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_H_CHn_EFR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**CSR** | RO | Channel Service Request<br>0b - Channel has no service request<br>1b - Channel has service request |
| **30**<br>**CHSR** | RO | Channel Service Request to Host<br>0b - Channel has no service request to host<br>1b - Channel has service request to host |
| **9**<br>**EMRL1** | RO | Match Recognition Latch 1 Event Status<br>0b - Channel has no match1 event<br>1b - Channel has match1 event |
| **8**<br>**EMRL2** | RO | Match Recognition Latch 2 Event Status<br>0b - Channel has no match2 event<br>1b - Channel has match2 event |
| **7**<br>**ETDL1** | RO | Transition Detect Latch 1 Event Status<br>0b - Channel has no transition detect1 event<br>1b - Channel has transition detect1 event |
| **6**<br>**ETDL2** | RO | Transition Detect Latch 2 Event Status<br>0b - Channel has no transition detect2 event<br>1b - Channel has transition detect2 event |
| **5**<br>**MRLE1** | RO | Match Recognition Latch Enable 1 Status<br>0b - Channel match1 event disable<br>1b - Channel match1 event enable |
| **4**<br>**MRLE2** | RO | Match Recognition Latch Enable 2 Status<br>0b - Channel match2 event disable<br>1b - Channel match2 event enable |
| **3**<br>**MRL1** | RO | Match Recognition Latch 1 Status<br>0b - Channel has no match1 flag<br>1b - Channel has match1 flag |
| **2**<br>**MRL2** | RO | Match Recognition Latch 2 Status<br>0b - Channel has no match2 flag<br>1b - Channel has match2 flag |
| **1**<br>**TDL1** | RO | Transition Detect Latch 1 Status<br>0b - Channel has no transition detect1 flag<br>1b - Channel has transition detect1 flag |
| **0**<br>**TDL2** | RO | Transition Detect Latch 2 Status<br>0b - Channel has no transition detect2 flag<br>1b - Channel has transition detect2 flag |

## 56.7 TPU_E Register Definition

TPU_E base address : 4044_6000h

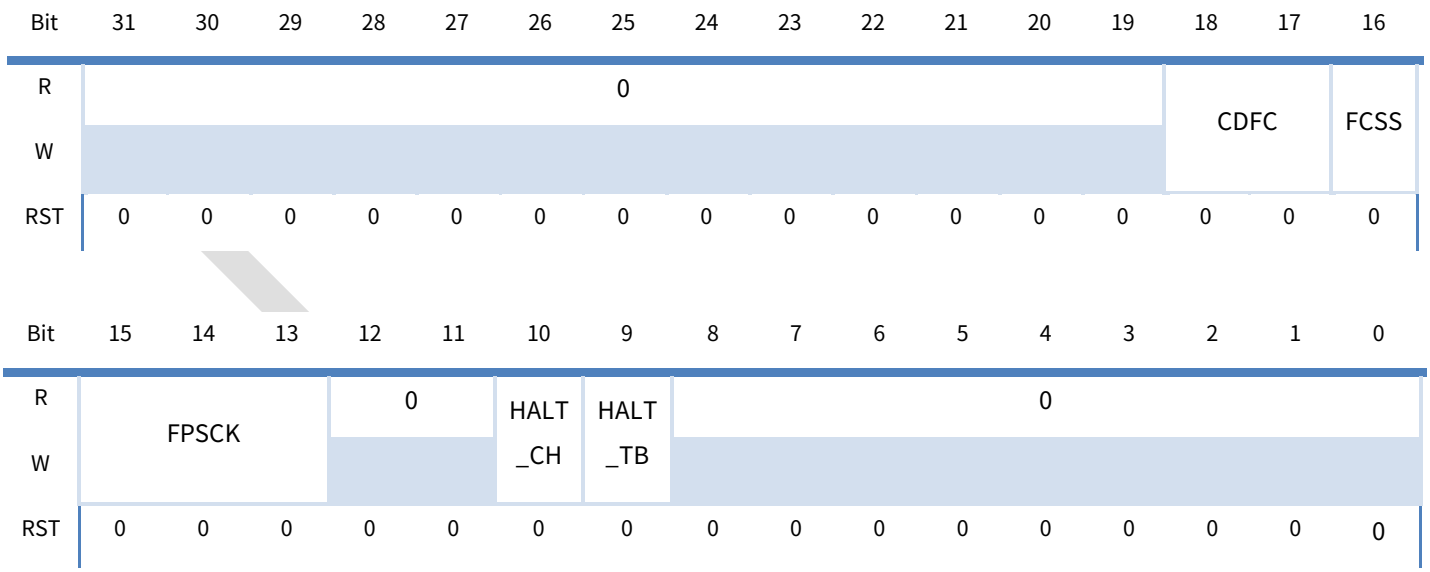
**Register Overview - TPU_E**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | TPU_E_GCR_CR (Global Control Region - Control Register) | 32 | RW | 0000_0000h | page |
| **8h** | TPU_E_GCR_SRR (Global Control Region - Software Reset Register) | 32 | W1C | 0000_0000h | page |
| **10h** | TPU_E_GCR_VSR (Global Control Region - Visible Status Register) | 32 | RO | 0000_0000h | page |
| **14h** | TPU_E_GCR_VIR (Global Control Region - Visible Input Status Register) | 32 | RO | 0000_0000h | page |
| **18h** | TPU_E_GCR_VOR (Global Control Region -Visible Output Status Register) | 32 | RO | 0000_0000h | page |
| **1Ch** | TPU_E_GCR_VOBR (Global Control Region - Visible Output Enable Status Register) | 32 | RO | 0000_0000h | page |
| **20h** | TPU_E_GCR_VM1R (Global Control Region - Visible MRL1 Register) | 32 | RO | 0000_0000h | page |
| **24h** | TPU_E_GCR_VM2R (Global Control Region - Visible MRL2 Register) | 32 | RO | 0000_0000h | page |
| **28h** | TPU_E_GCR_VT1R (Global Control Region - Visible TDL1 Register) | 32 | RO | 0000_0000h | page |
| **2Ch** | TPU_E_GCR_VT2R (Global Control Region - Visible TDL2 Register) | 32 | RO | 0000_0000h | page |
| **30h** | TPU_E_GCR_EM1R (Global Control Region - Visible Event by MRL1 Register) | 32 | RO | 0000_0000h | page |
| **34h** | TPU_E_GCR_EM2R (Global Control Region - Visible Event by MRL2 Register) | 32 | RO | 0000_0000h | page |
| **38h** | TPU_E_GCR_ET1R (Global Control Region - Visible Event by TDL1 Register) | 32 | RO | 0000_0000h | page |
| **3Ch** | TPU_E_GCR_ET2R (Global Control Region - Visible Event by TDL2 Register) | 32 | RO | 0000_0000h | page |
| **40h** | TPU_E_GCR_VHSR (Global Control Region - Visible Host Request Register) | 32 | RO | 0000_0000h | page |
| **F0h** | TPU_E_GCR_AFGR (Global Control Region - Interrupt Arbiter Fliter Grade Register) | 32 | RW | 0000_3B7Bh | page |
| **F4h** | TPU_E_GCR_ASR (Global Control Region - Interrupt Arbiter Status Register) | 32 | RW | 0000_0000h | page |
| **F8h** | TPU_E_GCR_AID (Global Control Region - Interrupt Arbiter ID Register) | 32 | RW | 0000_0C00h | page |

| | | | | | |
|---|---|---|---|---|---|
| **100h** | TPU_E_TBR_CR (Time Bases Region - Control Register) | 32 | RW | 0000_0000h | page |
| **104h** | TPU_E_TBR_T1R (Time Base Region - TCR1 Value Register) | 32 | RW | 0000_0000h | page |
| **108h** | TPU_E_TBR_T2R (Time Base Region - TCR2 Value Register) | 32 | RW | 0000_0000h | page |
| **110h** | TPU_E_TBR_TPR (Time Base Region - Tooth Program Register) | 32 | RW | 0000_0000h | page |
| **118h** | TPU_E_TBR_TRR (Time Base Region - Tick Rate Register) | 32 | RW | 0000_0000h | page |
| **11Ch** | TPU_E_TBR_T1MR (Time Base Region - TCR1 Maximum Register) | 32 | RW | 00FF_FFFFh | page |
| **114h** | TPU_E_TBR_T2MR (Time Base Region - TCR2 Maximum Register) | 32 | RW | 00FF_FFFFh | page |
| **800h+n*40h** | TPU_E_CHn_CR (Channel N Control Register) | 32 | RW | 0000_0000h | page |
| **804h+n*40h** | TPU_E_CHn_SR (Channel N Status Register) | 32 | RO | 0000_0000h | page |
| **808h+n*40h** | TPU_E_CHn_SCR (Channel N Status Control Register) | 32 | W1C | 0000_0000h | page |
| **80Ch+n*40h** | TPU_E_CHn_EFR (Channel N Event and Flag Register) | 32 | RO | 0000_0000h | page |
| **810h+n*40h** | TPU_E_CHn_CR2 (Channel N Control Register 2) | 32 | RW | 0000_0000h | page |
| **814h+n*40h** | TPU_E_CHn_MR (Channel N Mode configuration visible Register) | 32 | RW | 400C_0000h | page |
| **818h+n*40h** | TPU_E_CHn_ER1 (Channel N Event Control Register or Part 1) | 32 | RW | 0000_0000h | page |
| **81Ch+n*40h** | TPU_E_CHn_ER2 (Channel N Event Control Register or Part 2) | 32 | RW | 0000_0000h | page |
| **820h+n*40h** | TPU_E_CHn_ECR (Channel N Event Clear Register) | 32 | W1C | 0000_0000h | page |
| **824h+n*40h** | TPU_E_CHn_OCR (Channel N Output User Control Register) | 32 | RW | 0000_0000h | page |

### 56.7.1 TPU_E_GCR_CR (Global Control Region - Control Register)

Offset : 0h




**TPU_E_GCR_CR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **18-17**<br>**CDFC** | RW | Channel Digital Filter Control<br>These bits select a digital filtering mode for the channels when configured as inputs for improved noise immunity. The TPU has three digital filtering modes for the channels which provide programmable trade-off between signal latency and noise immunity. Changing CDFC during normal input channel operation is not recommended.<br>00b - Two Sample Mode. Using the filter clock which is the system clock divided by (2,4,...,256) as a sampling clock, comparing two consecutive samples which agree with each other sets the input signal state.<br>01b - Bypass Mode. The input signal is taken unfiltered, also making the channels work on T2/T4 timing mode.<br>10b - Three Sample Mode. Similar to two sample mode, but comparing three consecutive samples which agree with each other sets the input signal state.<br>11b - Continuous Mode. signal need to be stable for the whole filter clock period. This mode compares all the values at the rate of TPU clock (FCSS=1) or a TPU clock divided by two (FCSS =0), between two consecutive filter clock pulses. Signal needs to be continously stable for the entire period. If all the values agree with each other. Input signal state is updated. |
| **16**<br>**FCSS** | RW | Filter Clock Source Selection<br>Speeds up the filter clock source before the prescaler, allowing more input capture resolution at minimum prescaling.<br>0b - Use TPU clock/2 as edf clock source before prescaler<br>1b - Use TPU clock as edf clock source before prescaler |
| **15-13**<br>**FPSCK** | RW | Filter Prescaler Clock Control<br>FPSSCK controls the prescaling of the clocks used in digital filters for the channel input signals and TCRCLK input.<br>A new value written to FPSCK only becomes effective when the filter prescaler finishes the current count.<br>000 - Sample on system clock divided by 1<br>001 - Sample on system clock divided by 2<br>010 - Sample on system clock divided by 4<br>011 - Sample on system clock divided by 8<br>100 - Sample on system clock divided by 16<br>101 - Sample on system clock divided by 32<br>110 - Sample on system clock divided by 64<br>111 - Sample on system clock divided by 128 |
| **10**<br>**HALT_CH** | RW | Halt Channel array in Stop mode<br>0b - Channel function is normal in stop mode<br>1b - Channel functionality is disabled in stop mode |

| | | |
|---|---|---|
| **9**<br>**HALT_TB** | RW | Halt Time Bases Counter in stop mode<br>0b - Time bases counter run in stop mode<br>1b - Time base counter stop in stop mode |

### 56.7.2 TPU_E_GCR_SRR (Global Control Region - Software Reset Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | 0 |
| W | | | | | | | | | | | | | | | | SW_RST |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_SRR field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**SW_RST** | WO | Write FC005AFE to trigger a reset |

### 56.7.3 TPU_E_GCR_VSR (Global Control Region - Visible Status Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_VSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**SR** | RO | Each bit indicates the status of a channel.<br>For example, bit 31 indicates the channel 31's service request status. |

### 56.7.4 TPU_E_GCR_VIR (Global Control Region - Visible Input Status Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PSTI | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PSTI | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_VIR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**PSTI** | RO | Each bit indicates the status of a channel.<br>For example, bit 31 indicates the channel 31's pin input status. |

### 56.7.5 TPU_E_GCR_VOR (Global Control Region -Visible Output Status Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PSTO | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PSTO | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_VOR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**PSTO** | RO | Each bit indicates the status of a channel.<br>For example, bit 31 indicates the channel 31's pin output status. |

### 56.7.6 TPU_E_GCR_VOBR (Global Control Region - Visible Output Enable Status Register)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PSTOE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PSTOE | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_VOBR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**PSTOE** | RO | Each bit indicates the status of a channel.<br>For example, bit 31 indicates the channel 31's pin output enable status. |

### 56.7.7 TPU_E_GCR_VM1R (Global Control Region - Visible MRL1 Register)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | M1R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | M1R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_VM1R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0<br>M1R** | RO | Each bit indicates the status of a channel.<br>For example, bit 31 indicates the channel 31's MRL1. |

### 56.7.8 TPU_E_GCR_VM2R (Global Control Region - Visible MRL2 Register)

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | M2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | M2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_VM2R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0<br>M2R** | RO | Each bit indicates the status of a channel.<br>For example, bit 31 indicates the channel 31's MRL2. |

### 56.7.9 TPU_E_GCR_VT1R (Global Control Region - Visible TDL1 Register)

Offset : 28h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | T1R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | T1R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_VT1R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**T1R** | RO | Each bit indicates the status of a channel.<br>For example, bit 31 indicates the channel 31's TDL1. |

### 56.7.10 TPU_E_GCR_VT2R (Global Control Region - Visible TDL2 Register)

Offset : 2Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | T2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | T2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_VT2R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**T2R** | RO | Each bit indicates the status of a channel.<br>For example, bit 31 indicates the channel 31's TDL2. |

### 56.7.11 TPU_E_GCR_EM1R (Global Control Region - Visible Event by MRL1 Register)

Offset : 30h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EM1R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EM1R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_EM1R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**EM1R** | RO | Each bit indicates the status of a channel.<br>For example, bit 31 indicates the channel 31's event by MRL1. |

### 56.7.12 TPU_E_GCR_EM2R (Global Control Region - Visible Event by MRL2 Register)

Offset : 34h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EM2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EM2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_EM2R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**EM2R** | RO | Each bit indicates the status of a channel.<br>For example, bit 31 indicates the channel 31's event by MRL2. |

### 56.7.13 TPU_E_GCR_ET1R (Global Control Region - Visible Event by TDL1 Register)

Offset : 38h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ET1R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ET1R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_ET1R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0<br>ET1R** | RO | Each bit indicates the status of a channel.<br>For example, bit 31 indicates the channel 31's event by TDL1. |

### 56.7.14 TPU_E_GCR_ET2R (Global Control Region - Visible Event by TDL2 Register)

Offset : 3Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ET2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ET2R | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_ET2R field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0<br>ET2R** | RO | Each bit indicates the status of a channel.<br>For example, bit 31 indicates the channel 31's event by TDL2. |

### 56.7.15 TPU_E_GCR_VHSR (Global Control Region - Visible Host Request Register)

Offset : 40h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | HSR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | HSR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_VHSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 HSR** | RO | Each bit indicates the status of a channel.<br>For example, bit 31 indicates the channel 31's HSR. |

### 56.7.16 TPU_E_GCR_AFGR (Global Control Region - Interrupt Arbiter Fliter Grade Register)

Offset : F0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | FLT_GRD | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

**TPU_E_GCR_AFGR field descriptions**

| Field | Type | Description |
|---|---|---|
| **13-0 FLT_GRD** | RW | Filter Grade for 7 value in 1 loop.<br>00b - Disable<br>01b - Low<br>10b - Middle<br>11b - High |

### 56.7.17 TPU_E_GCR_ASR (Global Control Region - Interrupt Arbiter Status Register)

Offset : F4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ERROR | 0 | | | | | | | | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | ENDIAN |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_ASR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**ERROR** | W1C | Error Flag<br>0b - Configure filter grade successfully<br>1b - Failed to configure filter grade |
| **0**<br>**ENDIAN** | RW | Endian Configuration<br>0b - Big endian<br>1b - Little endian |

### 56.7.18 TPU_E_GCR_AID (Global Control Region - Interrupt Arbiter ID Register)

Offset : F8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | ETCS |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ETCS | | | | COND | | | | | 0 | ID | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_GCR_AID field descriptions**

| Field | Type | Description |
|---|---|---|
| **16-12**<br>**ETCS** | RO | Channel Function Entry Table Select<br>Entry function select.<br>0 - Function0<br>1 - Function1<br>…<br>31 - Function31 |
| **11-7**<br>**COND** | RO | Channel Function Entry Table Condition<br>Entry function condition, please refer table in the manual. |

| | | |
|---|---|---|
| **5-0**<br>**ID** | RO | Channel ID Info<br>0 - Channel 0<br>1 - Channel 1<br>…<br>31 - Channel 31<br>32 - Idle<br>42 - Error channel<br>63 - Configure filter grade is failed |

### 56.7.19 TPU_E_TBR_CR (Time Bases Region - Control Register)

Offset : 100h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TCR1P | | | | | | | | 0 | | | | | TCR1CTL | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TCR2P | | | | | | 0 | | ATGC | AM | | TCRCF | | TCR2CTL | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_TBR_CR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**TCR1P** | RW | TCR1 Prescaler Control.<br>TCR1 is clocked from the output of a prescaler. The prescaler divides its input by (TCR1P + 1) allowing frequency divisions from 1 to 256. The prescaler input is the TPU clock divided by 2, TPU clock or output of TCRCLK filtered. |
| **18-16**<br>**TCR1CTL** | RW | TCR1 Clock/Gate Control.<br>TCR1CTL = 3'b011 was updown counter mode<br>000b - Select TCRCLK as the clock source for the tcr1 prescaler<br>011b - TCR1 updown counter mode<br>100b - Select TPU clock divided by 2 as the clock source for the TCR1 prescaler<br>101b - Select TPU clock as the clock source for the TCR1 prescaler |

| | | |
|---|---|---|
| **15-10**<br>**TCR2P** | RW | TCR2 Prescaler Control.<br>TCR2 is clocked from the output of a prescaler. The prescaler divides its input by (TCR2P + 1) allowing frequency divisions from 1 to 64. The prescaler input is the TPU clock divided by 8, or Internal Timebase input, or TCRCLK filtered input. No effect on Angle mode. |
| **7**<br>**ATGC** | RW | Angle Tick Generator Clock.<br>0b - TCR1 prescaler output provides time intervals for tick generation,<br>1b - TCR2 prescaler output provides time intervals for tick generation. |
| **6-5**<br>**AM** | RW | Angle Mode Selection<br>00b - TCR2 was timebase (EAC disabled). Tooth signal and tooth processing channel was not applicable.<br>01b - TCR2 was angle ticks, Tooth signal was TCRCLK input and tooth processing channel was 0.<br>10b - TCR2 was angle ticks, Tooth signal was Channel 1 input and tooth processing channel was 1.<br>11b - TCR2 was angle ticks, Tooth signal was Channel 2 input and tooth processing channel was 2. |
| **4-3**<br>**TCRCF** | RW | TCRCLK Signal Filter Control<br>This field controls the TCRCLK digital filter, determining whether the TCRCLK signal input (after a synchronizer) is filtered with the same filter clock as the channel input signals or uses the TPU clock divided by 2.<br>00b - Two sample mode, Filter clock uses TPU clock divided by 2.<br>01b - Two sample mode, Filter clock uses the Filter clock of the channels.<br>10b - Integrator, Filter clock uses TPU clock divided by 2.<br>11b - Integrator, Filter clock uses the Filter clock of the channels. |

Flagchip

| | | |
|---|---|---|
| **2-0**<br>**TCR2CTL** | RW | TCR2 Clock/Gate Control<br>These bits are part of the TRC2 clock system. They determine the clock source for TCr2 before the prescaler. TCR2 can count on any detected edge of the TCRCLK signal or use it for gating TPU clock divided by 8. TCR2CTL also determines the TCRCLK edge selected for angle tooth detection in angle mode.<br>When AM=0, TCR2 clock before prescaler depends on TCR2CTL:<br>000 Gated DIV 8 clock. In this case, when the external TCRCLK signal is low, TCR2 prescaler is incremented at the frequency of the TPU clock divided by 8.<br>001b - Rise transition on TCRCLK signal increments the TRC2 prescaler.<br>010b - Fail transition on TCRCLK signal increments the TRC2 prescaler.<br>100b - DIV8 clock<br>TCR2CTL = 3'b101 was updown counter mode<br>111b - TCR2 frozen, except as STAC client.<br>When AM=1, TCR2 was used for angle tooth detection.<br>001b - Rising edge<br>010b - Failling edge<br>011b - Both edges<br>110b - no edge<br>For other conditions, do not use. |

### 56.7.20 TPU_E_TBR_T1R (Time Base Region - TCR1 Value Register)

Offset : 104h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TCR1 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TCR1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_TBR_T1R field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-0**<br>**TCR1** | RO | TCR1 Counter Value |

### 56.7.21 TPU_E_TBR_T2R (Time Base Region - TCR2 Value Register)

Offset : 108h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TCR2 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TCR2 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_TBR_T2R field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-0 TCR2** | RO | TCR2 Counter Value |

### 56.7.22 TPU_E_TBR_TPR (Time Base Region - Tooth Program Register)

Offset : 110h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LAST | MISSCNT | | IPH | HOLD | TPR1<br>0 | TICKS | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_TBR_TPR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **15**<br>**LAST** | RW | Last Tooth Indication<br>0b - Not Last Tooth.<br>1b - Last Tooth: reset TCR2 on the next physical tooth edge (or when IPH = 1) when MISSCNT = 0. |
| **14-13**<br>**MISSCNT** | RW | Missing Tooth Counter<br>00b - Not a Missing Tooth.<br>01b - One Missing Tooth.<br>10b - Two Missing Tooth.<br>11b - Three Missing Tooth. |
| **12**<br>**IPH** | RW | Insert Physical Tooth<br>This bit is used for exiting halt mode which is caused by missing a detection of a physical tooth. It generates a dummy physical tooth which has the same effect as a real physical tooth, and resets itself subsequently. If EAC is in halt mode, it switches back to normal mode. If EAC is in normal mode, it switches to high rate mode.<br>0b - No Operation.<br>1b - Insert dummy physical tooth. |
| **11**<br>**HOLD** | RW | Force EAC Halt<br>0b - Normal Operation.<br>1b - Force EAC to halt until detection of a physical tooth. |
| **10**<br>**TPR10** | RW | Reserved in Angle Mode. |
| **9-0**<br>**TICKS** | RW | Angle Ticks Number in the Current Tooth. |

### 56.7.23 TPU_E_TBR_TRR (Time Base Region - Tick Rate Register)

Offset : 118h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | INTEGER | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INTEGER | | | | | | | | FRACTION | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_TBR_TRR field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-9**<br>**INTEGER** | RW | The integer part of TCR1 clocks in one Angle tick.<br>This number, decremented by one, is a down-counter preload value. A value of INTEGER=0 represents an integer of 32768. |
| **8-0**<br>**FRACTION** | RW | Nine-bit fractional part of TCR1 clocks in one angle tick.<br>The FRACTION value is accumulated in the EAC Fraction Accumulator, and whenever the result overflows. The tick prescaler is halted for one TCR1 clock. |

### 56.7.24 TPU_E_TBR_T1MR (Time Base Region - TCR1 Maximum Register)

Offset : 11Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OVF | 0 | | | IRQ_ | 0 | | | MAX | | | | | | | |
| W | w1c | | | | EN | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAX | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**TPU_E_TBR_T1MR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**OVF** | W1C | TCR1 Overflow Flag<br>W1C.<br>0b - No overflow flag<br>1b - Overflow flag |
| **27**<br>**IRQ_EN** | RW | TCR1 Overflow IRQ Enable<br>0b - Overflow IRQ disable<br>1b - Overflow IRQ enable |
| **23-0**<br>**MAX** | RW | Maximum value of TCR1 counter in TCR1 updown mode |

### 56.7.25 TPU_E_TBR_T2MR (Time Base Region - TCR2 Maximum Register)

Offset : 114h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OVF | 0 | | | IRQ_EN | 0 | | | MAX | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAX | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**TPU_E_TBR_T2MR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**OVF** | W1C | TCR2 Overflow Flag<br>W1C.<br>0b - No overflow flag<br>1b - Overflow flag |
| **27**<br>**IRQ_EN** | RW | TCR2 Overflow IRQ Enable<br>0b - Overflow IRQ disable<br>1b - Overflow IRQ enable |
| **23-0**<br>**MAX** | RW | Maximum value of TCR2 counter in TCR2 updown mode |

### 56.7.26 TPU_E_CHn_CR (Channel N Control Register)

Offset : 800h+N*40h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CIE | CPR | | CFB | CFS | | | | | ETPD | ECTS | FLG1 | FLG0 | 0 | OPOL | ODIS |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | CHSA | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_CHn_CR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**CIE** | RW | Channel Event Triggered Interrupt Enable<br>The corresponding clear bit CHn_SCR[CISC] can not clear such IRQ, but can clear tunnel on irq_arbiter.<br>0b - Disable<br>1b - Enable |
| **30-29**<br>**CPR** | RW | Channel Event Triggered Interrupt Priority<br>00b - Disabled<br>01b - Low<br>10b - Middle<br>11b - High |
| **28**<br>**CFB** | RW | Channel Filter Bypass<br>0b - Channel filer open<br>1b - Channel filter bypass |
| **27-23**<br>**CFS** | RW | Channel Entry Function Select<br>Select entry function, for example, 5'b00000 is function0. |
| **22**<br>**ETPD** | RW | Channel Function Entry Table Pin Direcion<br>0b - Use PSTI for entry point selection<br>1b - Use PSTO for entry point selection |
| **21**<br>**ECTS** | RW | Channel Function Entry Table Condition Select<br>0b - Select standard entry table condition encoding scheme<br>1b - Select alternate entry table condition encoding scheme |
| **20**<br>**FLG1** | RW | Channel Function Entry Table Flag 1<br>user configurable |
| **19**<br>**FLG0** | RW | Channel Function Entry Table Flag 0<br>User configurable |
| **17**<br>**OPOL** | RW | Output Polarity<br>0b - Output active low (output disable drives output to high).<br>1b - Output active high (output disable drives output to low). |
| **16**<br>**ODIS** | RW | Output Disable<br>0b - Turns off the output disable feature for the channel.<br>1b - Turns on the output disable feature for the channel. |
| **2-0**<br>**CHSA** | RW | Channel HSA index<br>0x00 - No HSA<br>Others - Has HSA |

### 56.7.27 TPU_E_CHn_SR (Channel N Status Register)

Offset : 804h+N*40h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CIS | 0 | | | | | | CHRS | 0 | | | | | CHRI | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CEIS | CHDS | 0 | CTS | 0 | | | CHAS | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_CHn_SR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**CIS** | RO | Channel Event Trigger Interrupt Status<br>0b - No event trigger interrupt<br>1b - Event trigger interrupt |
| **24**<br>**CHRS** | RO | Channel Host Service Request (HSR) Status<br>0b - No HSR<br>1b - HSR generated |
| **18-16**<br>**CHRI** | RO | Channel HSR Info & Index<br>Index was 18:16.<br>000b - No HSR<br>Others - Has HSR |
| **15**<br>**CEIS** | RO | Channel Event as Interrupt Status<br>0b - Channel has no interrupt request by event<br>1b - Channel has interrupt request by request |
| **14**<br>**CHDS** | RO | Channel Host Service Acknowledge (HSA) DMA Request Status<br>0b - Channel has no DMA request<br>1b - Channel has DMA request |
| **12**<br>**CTS** | RO | Channel Trigger Status<br>0b - Channel has no trigger request<br>1b - Channel has trigger request |
| **8**<br>**CHAS** | RO | Channel Host Service Acknowledge (HSA) Interrupt Status<br>0b - Channel has no HSA interrupt request<br>1b - Channel has HAS interrupt request |

### 56.7.28 TPU_E_CHn_SCR (Channel N Status Control Register)

Offset : 808h+N*40h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CISC | CHDT | 0 | | | | | CHRC | 0 | | | | | | | CHAT |
| W | w1c | w1c | | | | | | w1c | | | | | | | | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_CHn_SCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**CISC** | W1C | Channel Event Trigger Interrupt Tunnel Clear Status<br>Cannot clear interrupt itself.<br>Write 1 to clear interrupt status. |
| **30**<br>**CHDT** | W1C | Channel Host Service Acknowledge (HSA) DMA Request Trigger<br>Write 1 to clear DMA request status. |
| **24**<br>**CHRC** | W1C | Channel HSR Clear<br>Write 1 to clear HSR. |
| **16**<br>**CHAT** | W1C | Channel Host Service Acknowledge (HSA) Interrupt Trigger<br>Write 1 to clear HSA. |

### 56.7.29 TPU_E_CHn_EFR (Channel N Event and Flag Register)

Offset : 80Ch+N*40h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CSR | CHSR | 0 | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | EMRL1 | EMRL2 | ETDL1 | ETDL2 | MRLE1 | MRLE2 | MRL1 | MRL2 | TDL1 | TDL2 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_CHn_EFR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**CSR** | RO | Channel Service Request<br>0b - Channel has no service request<br>1b - Channel has service request |
| **30**<br>**CHSR** | RO | Channel Service Request to Host<br>0b - Channel has no service request to host<br>1b - Channel has service request to host |
| **9**<br>**EMRL1** | RO | Match Recognition Latch 1 Event status<br>0b - Channel has no match1 event<br>1b - Channel has match1 event |
| **8**<br>**EMRL2** | RO | Match Recognition Latch 2 Event status<br>0b - Channel has no match2 event<br>1b - Channel has match2 event |
| **7**<br>**ETDL1** | RO | Transition Detect Latch 1 Event status<br>0b - Channel has no transition detect1 event<br>1b - Channel has transition detect1 event |
| **6**<br>**ETDL2** | RO | Transition Detect Latch 2 Event status<br>0b - Channel has no transition detect2 event<br>1b - Channel has transition detect2 event |
| **5**<br>**MRLE1** | RO | Match Recognition Latch Enable 1 status<br>0b - Channel match1 event disable<br>1b - Channel match1 event enable |
| **4**<br>**MRLE2** | RO | Match Recognition Latch Enable 2 status<br>0b - channel match2 event disable<br>1b - channel match2 event enable |
| **3**<br>**MRL1** | RO | Match Recognition Latch 1 status<br>0b - Channel has no match1 flag<br>1b - Channel has match1 flag |
| **2**<br>**MRL2** | RO | Match Recognition Latch 2 status<br>0b - Channel has no match2 flag<br>1b - Channel has match2 flag |
| **1**<br>**TDL1** | RO | Transition Detect Latch 1 status<br>0b - Channel has no transition detect1 flag<br>1b - Channel has transition detect1 flag |

| | | |
|---|---|---|
| **0**<br>**TDL2** | RO | Transition Detect Latch 2 status<br>0b - Channel has no transition detect2 flag<br>1b - Channel has transition detect2 flag |

### 56.7.30 TPU_E_CHn_CR2 (Channel N Control Register 2)

Offset : 810h+N*40h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TBS1 | | | PSTI | PSTO | PRSS | 0 | | | IPAC1 | | | OBE | OPAC1 | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TBS2 | | | 0 | | | | | | IPAC2 | | | 0 | OPAC2 | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_CHn_CR2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-29**<br>**TBS1** | RW | Time Base Selection of Part 1.<br>TBS[2] was comparator selection for Match Recognition.<br>0 - Greater or Equal.<br>1 - Equal Only.<br>TBS[1] was Capture time base selection.<br>0 - TCR1.<br>1 - TCR2.<br>TBS[0] was Match time base selection.<br>0 - TCR1.<br>1 - TCR2. |
| **28**<br>**PSTI** | RO | Pin State Input<br>This bit shows the current value of the channel input signal state |
| **27**<br>**PSTO** | RO | Pin State Output<br>This bit shows the current value of the filtered channel output signal state |
| **26**<br>**PRSS** | RO | Pin Request Service Sample<br>This bit shows the current value of the channel input request sample signal state |

| | | |
|---|---|---|
| **22-20**<br>**IPAC1** | RW | Input Pin Action Control of Part 1<br>000b - Do not detect transitions.<br>001b - Detect rising edge only.<br>010b - Detect falling edge only.<br>011b - Detect either edge.<br>100b - Detect input signal = 0 on Match.<br>101b - Detect input signal = 1 on Match.<br>Others - Reserved. |
| **19**<br>**OBE** | RW | Output Buffer Enable<br>0b - Output buffer disable<br>1b - Output buffer enable |
| **18-16**<br>**OPAC1** | RW | Output Pin Action Control of Part 1<br>000b - Do not change output signal.<br>001b - Match sets output high.<br>010b - Match sets output low.<br>011b - Match toggles output signal.<br>100b - Transition detection sets output high.<br>101b - Transition detection sets output low.<br>110b - Transition detection toggles output signal.<br>111b - Do not change OPAC |
| **15-13**<br>**TBS2** | RW | Time Base Selection of Part 2<br>TBS[2] was comparator selection for Match Recognition.<br>0 - Greater or Equal.<br>1 - Equal Only.<br>TBS[1] was Capture time base selection.<br>0 - TCR1.<br>1 - TCR2.<br>TBS[0] was Match time base selection.<br>0 - TCR1.<br>1 - TCR2. |
| **6-4**<br>**IPAC2** | RW | Input Pin Action Control of Part 2<br>000b - Do not detect transitions.<br>001b - Detect rising edge only.<br>010b - Detect falling edge only.<br>011b - Detect either edge.<br>100b - Detect input signal = 0 on Match.<br>101b - Detect input signal = 1 on Match.<br>Others - Reserved. |

| | | |
|---|---|---|
| **2-0**<br>**OPAC2** | RW | Output Pin Action Control of Part 2<br>000b - Do not change output signal.<br>001b - Match sets output high.<br>010b - Match sets output low.<br>011b - Match toggles output signal.<br>100b - Transition detection sets output high.<br>101b - Transition detection sets output low.<br>110b - Transition detection toggles output signal.<br>111b - Do not change OPAC |

### 56.7.31 TPU_E_CHn_MR (Channel N Mode configuration visible Register)

Offset : 814h+N*40h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SRIE | SRI | 0 | | | | | | | | | PDME | PDCM | | | |
| W | w1c | | | | | | | | | | | w1c | | | | |
| RST | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TCCE | UDME | UDCM | | | | | | | | | | | | | |
| W | | w1c | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_CHn_MR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**SRIE** | W1C | Sevice Request Write Enable<br>SRI will configure to Channel when SRIE = 1. |
| **30**<br>**SRI** | RW | Service Request Inhibit Latch<br>0b - Service request enable<br>1b - Service request disable |
| **20**<br>**PDME** | W1C | PDCM Write Enable<br>PDCM will configure to Channel when PDME = 1. |
| **19-16**<br>**PDCM** | RW | Predefined Channel Mode |

| | | |
|---|---|---|
| **15**<br>**TCCE** | RW | Transition Continuous Capture Enable<br>TCCE enables capture from transitions after TDL1 is set.<br>0b - TCCE mode disable<br>1b - TCCE mode enable |
| **14**<br>**UDME** | W1C | UDCM Write Enable<br>UDCM will configure to Channel when UDME = 1. |
| **13-0**<br>**UDCM** | RW | User Defined Channel Mode |

### 56.7.32 TPU_E_CHn_ER1 (Channel N Event Control Register or Part 1)

Offset : 818h+N*40h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ERW | 0 | | | | | MEF | ERS | ER1 | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ER1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_CHn_ER1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**ERW** | W1C | Match Configuration Enable<br>0b - Match value load disable<br>1b - Match value load enable |
| **25**<br>**MEF** | RW | Match Enable Flag<br>0b - Match disable<br>1b - Match enable |
| **24**<br>**ERS** | RW | Capture/Match Select for Value of ER1/ER2. |
| **23-0**<br>**ER1** | RW | ER1 Value<br>Valid only when ERW = 1. |

### 56.7.33 TPU_E_CHn_ER2 (Channel N Event Control Register or Part 2)

Offset : 81Ch+N*40h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ERW | 0 | | | | | | | ER2 | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ER2 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_CHn_ER2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**ERW** | W1C | Match Configuration Enable<br>0b - Match value load disable<br>1b - Match value load enable |
| **23-0**<br>**ER2** | RW | ER2 Value<br>Valid only when ERW = 1. |

### 56.7.34 TPU_E_CHn_ECR (Channel N Event Clear Register)

Offset : 820h+N*40h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MRE1_CLR | MRL1_CLR | TDL1_CLR | | | | | | | 0 | | | | | | |
| W | w1c | w1c | w1c | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MRE2_CLR | MRL2_CLR | TDL2_CLR | | | | | | | 0 | | | | | | |
| W | w1c | w1c | w1c | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_CHn_ECR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31**<br>**MRE1_CLR** | W1C | Match1 Configuration Enable Clear<br>Write 1 to clear the match1 configuration flag. |
| **30**<br>**MRL1_CLR** | W1C | Match1 Event Clear<br>Write 1 to clear the match1 event flag. |
| **29**<br>**TDL1_CLR** | W1C | Transition Detect 1 Event Clear<br>Write 1 to clear the transition detect1 event flag. |
| **15**<br>**MRE2_CLR** | W1C | Match2 Configuration Enable Clear<br>Write 1 to clear the match1 configuration flag. |
| **14**<br>**MRL2_CLR** | W1C | Match2 Event Clear<br>Write 1 to clear the match1 event flag. |
| **13**<br>**TDL2_CLR** | W1C | Transition Detect 2 Event Clear<br>Write 1 to clear the transition detect2 event flag. |

### 56.7.35 TPU_E_CHn_OCR (Channel N Output User Control Register)

Offset : 824h+N*40h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | OUT_HIS | OUT_OP1 | OUT_OP2 | 0 | | | | | | | | | |
| W | | | | w1c | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TPU_E_CHn_OCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **12**<br>**OUT_HIS** | W1C | Output User Control, High or Low Select<br>0b - Output select low<br>1b - Output select high |
| **11**<br>**OUT_OP1** | RW | Output User Control of Part 1<br>0b - Not select OPAC1<br>1b - Select OPAC1 |

| | | |
|---|---|---|
| **10**<br>**OUT_OP2** | RW | Output User Control of Part 2<br>0b - Not select OPAC2<br>1b - Select OPAC2 |

Flagchip

# Chapter 57 FC Serial Peripheral Interface (FCSPI)

## 57.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

This chapter describes the FC Serial Peripheral Interface (FCSPI) module in the following sections:

- Chip-specific Information
- Features
- Overview
- Functional Description
- Registers

## 57.2 Chip-specific Information

N/A

## 57.3 Features

The FCSPI has the following features:

- Support master mode and slave mode
- Support up to 4 Peripheral Chip Select (PCS) in master mode
- Support software reset
- Configurable PCS polarity
- Configurable Serial Clock (SCK) polarity and phase
- TX/RX FIFO size is 8 words
- Configurable SCK frequency (Refer to the Data Sheet for the maximum frequency)
- Configurable delay between the PCS and SCK edge
- Support full- and half-duplex transfers in 1-bit mode
- Support half-duplex transfers in 2-bit and 4-bit mode
- Can use a trigger to start an FCSPI bus transfer in master mode
- Support data match interrupts
- Each FCSPI can monitor other FCSPIs on the chip

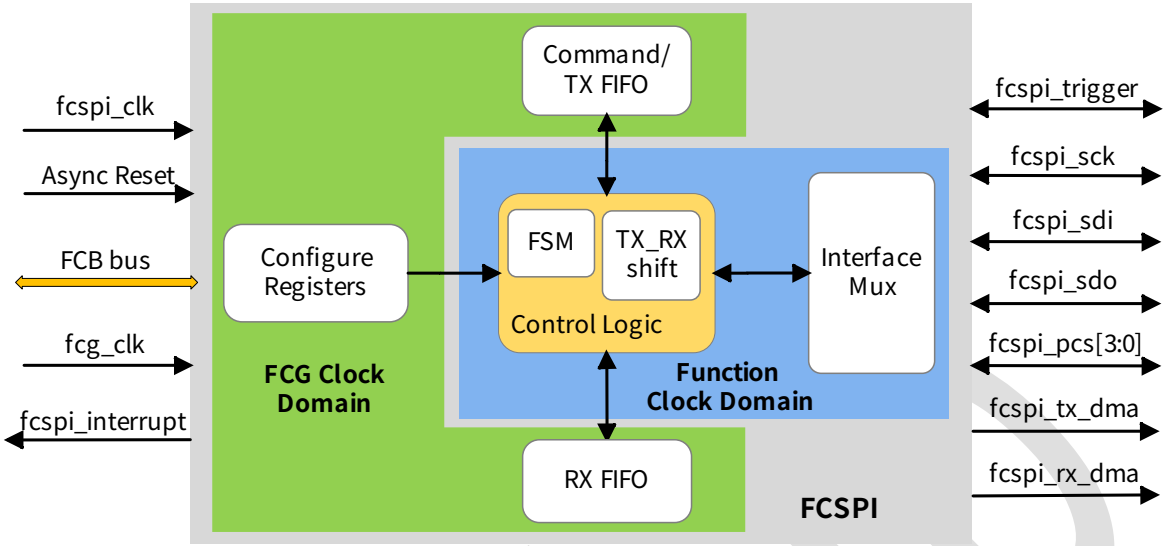
## 57.4 Overview

The FCSPI is a module that supports an interface to a Serial Peripheral Interface (SPI) bus as a master or a slave. All FCSPIs can work in stop mode and FCSPI0 can work in standby mode if a functional clock is available.

Below is the block diagram of FCSPI.

Flagchip

**Figure 136. FCSPI block diagram**




## 57.5 Functional Description

### 57.5.1 Software Reset

A software reset can reset the value of all registers in the FCSPI module and reset the internal logic as well. The software reset is asserted by setting CTRL[SW_RST] and negated by clearing CTRL[SW_RST].

Step 1: Generate software reset (CTRL[SW_RST] = 1).

Step 2: Clear software reset (CTRL[SW_RST] = 0).

### 57.5.2 New Transfer in Master Mode

The FCSPI supports two types of master transfer: normal transfer and trigger transfer. For normal transfer, the FCSPI will start a new transfer if the TX FIFO is not empty and FCSPI is enabled (and other configurations have been done). For trigger transfer, the FCSPI will start a new transfer when a trigger input comes after all configurations are done and TX FIFO is not empty.

#### *57.5.2.1 Start Normal Transfer*

Step 1: Configure FCSPI to master mode (CFG1[MASTER] = 1'b1).

Step 2: Write the TR_CTRL register with the specified configuration. For example:

- Use PCS1 as peripheral chip select (PCS=1)
- FCSPI input clock divided by 8 to generate functional clock (PRESCALE=3)
- Data bus width is 1-bit transfer (WIDTH=0)
- Frame size is 32 bits (FRM_SZ=31)

Step 3: Write the transmit data into TX FIFO.

Step 4: Enable FCSPI(CTRL[FCSPI_EN] = 1'b1), then FCSPI will transmit the data in TX FIFO and store the received data into RX FIFO.

#### *57.5.2.2 Start Trigger Transfer*

Step 1: Configure FCSPI to master mode (CFG1[MASTER] = 1'b1).

Step 2: Configure TRGSEL to select a trigger source (For detail, please refer to chapter TRGSEL)

Step 3: Enable FCSPI trigger (CFG0[TRGEN] = 1'b1).

Step 4: Write the TR_CTRL register with the specified configuration. For example:

- Use PCS1 as peripheral chip select (PCS=1)

- FCSPI input clock divided by 8 to generate functional clock (PRESCALE=3)
- Data bus width is 1-bit transfer (WIDTH=0)
- Frame size is 32 bits (FRM_SZ=31)

Step 5: Write the transmit data into TX FIFO.

Step 6: Enable FCSPI(CTRL[FCSPI_EN] = 1'b1).

Step 7: Wait for the trigger asserted. FCSPI will transmit the data in TX FIFO and store the received data into RX FIFO once the trigger is asserted.

### 57.5.3 New Transfer in Slave Mode

The FCSPI module supports slave mode. In slave mode, the PCS and SCK are generated externally by the master, that is the transfer is started by the master. Software needs to complete the configuration and fill the TX FIFO with transmitting data before the transfer start.

Step 1: Configure FCSPI to slave mode (CFG1[MASTER] = 1'b0).

Step 2: Write the TR_CTRL register with the specified configuration. For example:

- Use PCS0 as peripheral chip select (PCS=0)
- FCSPI input clock divided by 4 to generate functional clock (PRESCALE=2)
- Data bus width is 1-bit transfer (WIDTH=0)
- Frame size is 32 bits (FRM_SZ=31)

Step 3: Write the transmit data into TX FIFO.

Step 4: Enable FCSPI(CTRL[FCSPI_EN] = 1'b1)

Step 5: Wait for the master to start a transfer, then FCSPI will transmit the data in TX FIFO and store the received data into RX FIFO.

### 57.5.4 Parallel Transfer (in Master Mode)

The FCSPI module supports 2-bit and 4-bit parallel transfer. Both 2-bit and 4-bit transfers are half-duplex transfers. For 2-bit transfer, the FCSPI will use SIN and SOUT as data pins. For 4-bit transfer, the FCSPI will use SIN, SOUT, and PCS[3:2] as data pins. Here take 4-bit transfer as an example.

Step 1: Configure FCSPI to master mode (CFG1[MASTER] = 1'b1).

Step 2: Configure FCSPI data bus pins (CFG1[PCS_CFG]=1'b1, CFG1[OUT_CFG]=1'b1, CFG1[PIN_CFG]=2'b00).

Step 3: Write the TR_CTRL register with the specified configuration. For example:

- Use PCS0 as peripheral chip select (PCS=0)
- FCSPI input clock divided by 8 to generate functional clock (PRESCALE=3)
- Data bus width is 4-bit transfer (WIDTH=2)
- Frame size is 32 bits (FRM_SZ=31)

Step 4: Write the transmit data into TX FIFO.

Step 5: Enable FCSPI(CTRL[FCSPI_EN] = 1'b1), then FCSPI will transmit the data in TX FIFO and store the received data into RX FIFO.

### 57.5.5 Transfer Monitor

Each FCSPI module can monitor the transfer data of other FCSPI modules on the chip. The FCSPI monitor feature is intended to check if the data transmitted to the pin or received from the pin is correct which can improve the robustness of the FCSPI. The steps below describe how to use FCSPI0 to monitor FCSPI1.

Step 1: Configure FCSPI0 as a slave and set TX_MSK, other configurations of the frame are consistent with FCSPI1.

Step 2: Configure FCSPI0 to monitor FCSPI1 via writing PERIMUX0 register (set PERIMUX0[FCSPI0_MON] = 1).

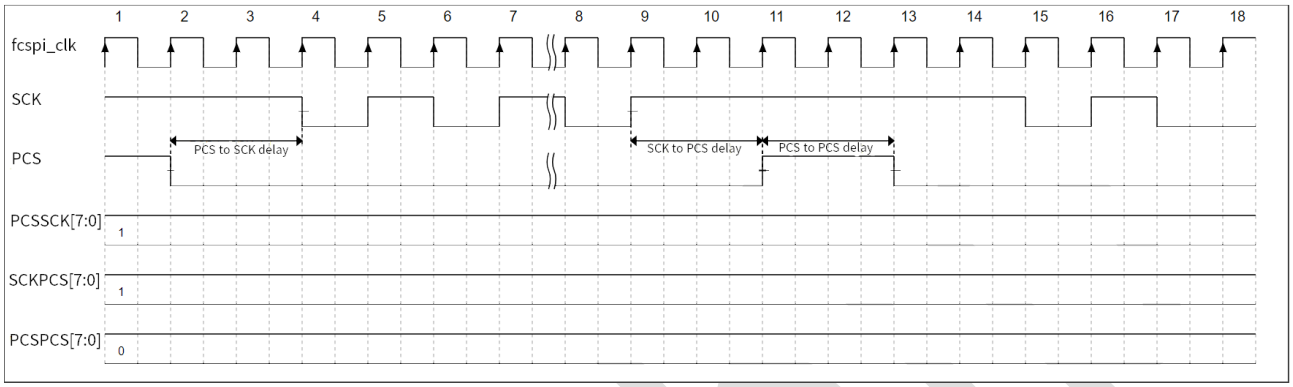
Step 3: Configure FCSPI1 to start a transfer in master mode.

Step 4: Compare the data received by the monitor with the transmitted data.

### 57.5.6 PCS-SCK Configuration

Software can configure the delay between PCS to SCK, SCK to PCS, and PCS to PCS. The waveform below helps to illustrate the delay cycles.

**Figure 137. Waveform of PCS-SCK configuration**




## 57.6 Register Definition

FCSPI0 base address : 4006_2000h

FCSPI1 base address : 4006_3000h

FCSPI2 base address : 4006_4000h

FCSPI3 base address : 4043_F000h

FCSPI4 base address : 4044_0000h

FCSPI5 base address : 4044_1000h

**Register Overview - FCSPI**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **10h** | FCSPI_CTRL (Control Register) | 32 | RW | 0000_0000h | page |
| **14h** | FCSPI_STATUS (Status Register) | 32 | W1C | 0000_0001h | page |
| **18h** | FCSPI_INT_EN (Interrupt Enable Register) | 32 | RW | 0000_0000h | page |
| **1Ch** | FCSPI_DMA_EN (DMA Enable Register) | 32 | RW | 0000_0000h | page |
| **20h** | FCSPI_CFG0 (Configuration Register 0) | 32 | RW | 0000_0004h | page |
| **24h** | FCSPI_CFG1 (Configuration Register 1) | 32 | RW | 0000_0000h | page |
| **30h** | FCSPI_DATA_MATCH0 (Data Match Register 0) | 32 | RW | 0000_0000h | page |
| **34h** | FCSPI_DATA_MATCH1 (Data Match Register 1) | 32 | RW | 0000_0000h | page |
| **40h** | FCSPI_CLK_CFG (Clock Configuration Register) | 32 | RW | 0000_0000h | page |
| **58h** | FCSPI_FIFO_WTM (FIFO Water Mark Register) | 32 | RW | 0000_0000h | page |
| **5Ch** | FCSPI_FIFO_STATUS (FIFO Status Register) | 32 | RO | 0000_0000h | page |
| **60h** | FCSPI_TR_CTRL (Transmit and Receive Control Register) | 32 | RW | 0000_001Fh | page |

| | | | | | |
|---|---|---|---|---|---|
| **64h** | FCSPI_TX_DATA (Transmit Data Register) | 32 | WO | 0000_0000h | page |
| **70h** | FCSPI_RX_STATUS (Receive Status Register) | 32 | RO | 0000_0002h | page |
| **74h** | FCSPI_RX_DATA (Receive Data Register) | 32 | RO | 0000_0000h | page |

### 57.6.1 FCSPI_CTRL (Control Register)

Software can reset or enable FCSPI module via writing this register.

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | 0 | 0 | 0 | | | | DBG_EN | WAIT_DIS | SW_RST | M_EN |
| W | | | | | | | RST_RF | RST_TF | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSPI_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **9**<br>**RST_RF** | WO | Reset Receive FIFO<br>0b - Do nothing<br>1b - Reset RX FIFO |
| **8**<br>**RST_TF** | WO | Reset Transmit FIFO<br>0b - Do nothing<br>1b - Reset TX FIFO |
| **3**<br>**DBG_EN** | RW | Debug Mode Enable<br>0b - Disable FCSPI in debug mode<br>1b - Enable FCSPI in debug mode |
| **2**<br>**WAIT_DIS** | RW | WAIT Mode Disable<br>0b - Enable FCSPI in wait mode<br>1b - Disable FCSPI in wait mode |
| **1**<br>**SW_RST** | RW | Software Reset<br>0b - Do nothing<br>1b - Reset FCSPI |

| | | |
|---|---|---|
| **0**<br>**M_EN** | RW | Module Enable<br>0b - Disable FCSPI<br>1b - Enable FCSPI |

### 57.6.2 FCSPI_STATUS (Status Register)

Software can get the status of FCSPI transfer via reading this register.

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | BF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DMF | RX_FO | TX_FU | TCF | FEF | RX_WF | 0 | | | | | | RX_FF | TX_FF |
| W | | | w1c | w1c | w1c | w1c | w1c | w1c | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**FCSPI_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **24**<br>**BF** | RO | Busy Flag<br>0b - FCSPI is idle<br>1b - FCSPI is busy |
| **13**<br>**DMF** | W1C | Data Match Flag<br>0b - RX data match not detected<br>1b - RX data match detected |
| **12**<br>**RX_FO** | W1C | RX FIFO Overflow<br>0b - RX FIFO has not overflowed<br>1b - RX FIFO has overflowed |
| **11**<br>**TX_FU** | W1C | TX FIFO Underrun<br>0b - Transmit FIFO underrun has not occurred<br>1b - Transmit FIFO underrun has occurred |
| **10**<br>**TCF** | W1C | Transfer Complete Flag<br>0b - Transfers has not completed<br>1b - Transfer has completed and FCSPI goes to idle status |

| | | |
|---|---|---|
| **9**<br>**FEF** | W1C | Frame End Flag<br>0b - Frame end not detected<br>1b - Frame end detected |
| **8**<br>**RX_WF** | W1C | Receive Word Flag<br>0b - Word has not been received<br>1b - Word has been received |
| **1**<br>**RX_FF** | RO | Receive FIFO Flag<br>0b - RX Data is not greater than FCR[RXWATER] (FIFO Control Register)<br>1b - RX data is greater than FCR[RXWATER] |
| **0**<br>**TX_FF** | RO | Transmit FIFO Flag<br>0b - Transmit data is more than FCR[TXWATER] (FIFO Control Register)<br>1b - Transmit data is equal to or less than FCR[TXWATER] |

### 57.6.3 FCSPI_INT_EN (Interrupt Enable Register)

Software can enable the interrupt of FCSPI via writing this register.

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DMIE | RFOIE | TFUIE | TCIE | FEIE | RWIE | 0 | | | | | | RFIE | TFIE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSPI_INT_EN field descriptions**

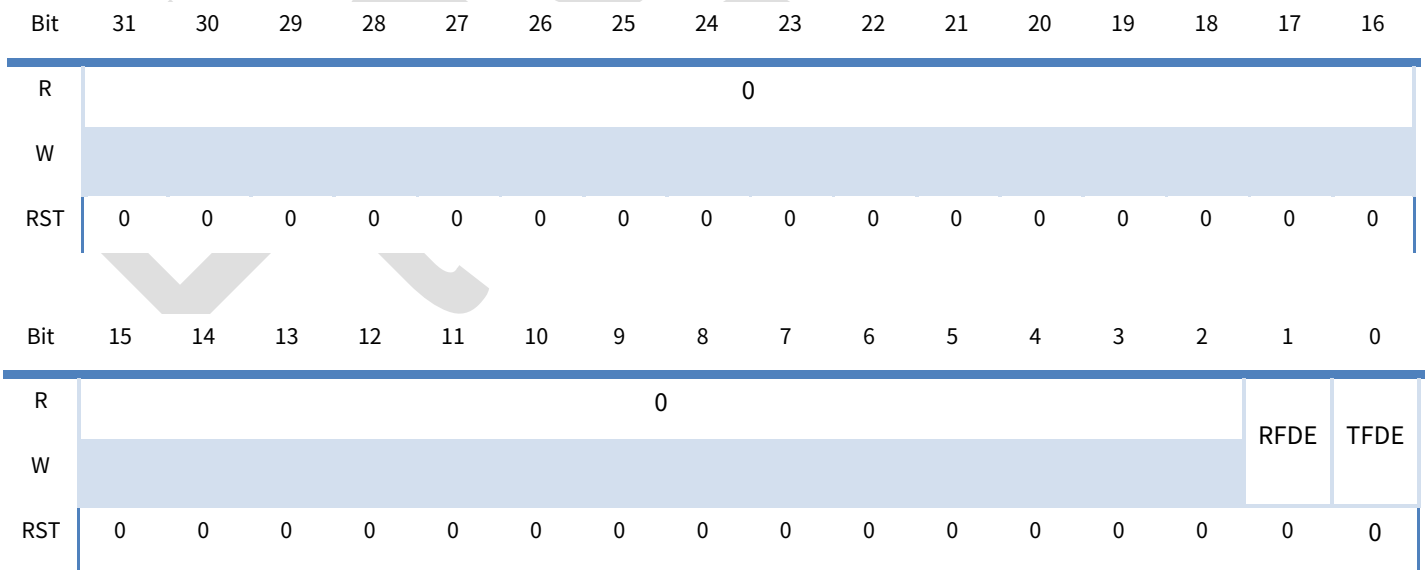
| Field | Type | Description |
|---|---|---|
| **13**<br>**DMIE** | RW | Data Match Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **12**<br>**RFOIE** | RW | RX FIFO Overflow Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

| | | |
|---|---|---|
| **11**<br>**TFUIE** | RW | TX FIFO Underrun Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **10**<br>**TCIE** | RW | Transfer Complete Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **9**<br>**FEIE** | RW | Frame End Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **8**<br>**RWIE** | RW | RX Word Complete Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **1**<br>**RFIE** | RW | RX FIFO Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **0**<br>**TFIE** | RW | TX FIFO Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

### 57.6.4 FCSPI_DMA_EN (DMA Enable Register)

Software can enable the DMA request of FCSPI via writing this register.

Offset : 1Ch




**FCSPI_DMA_EN field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **1**<br>**RFDE** | RW | RX FIFO DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |
| **0**<br>**TFDE** | RW | TX FIFO DMA Enable<br>0b - DMA request is disabled<br>1b - DMA request is enabled |

### 57.6.5 FCSPI_CFG0 (Configuration Register 0)

This is a configuration register.

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | AKDS | RDMO | 0 | | | | | | | | TRGEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**FCSPI_CFG0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **10**<br>**AKDS** | RW | Ack Mode Disable<br>In slave mode (and CT_EN=1, AKDS=0), the FCSPI will only transmit the first FRAMESZ bits, after which the FCSPI will transmit received data until the next PCS negation.<br>0b - FCSPI works in ack mode when configured as slave continuous mode.<br>1b - FCSPI ack mode is disabled when configured as slave continuous mode. |
| **9**<br>**RDMO** | RW | Receive Data Match Only<br>0b - Receive and store all data in RX FIFO<br>1b - Receive and store data after Data Match Flag set |
| **0**<br>**TRGEN** | RW | Trigger Enable<br>0b - Trigger disabled<br>1b - Trigger can start a FCSPI transfer |

### 57.6.6 FCSPI_CFG1 (Configuration Register 1)

This is a configuration register.

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | PCS_CFG | OUT_CFG | PIN_CFG | | 0 | | | | | MAT_CFG | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | PCS_POL | | | | 0 | | | | | INT_PCS | SCK_LB | MASTER |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSPI_CFG1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **27**<br>**PCS_CFG** | RW | PCS Configuration<br>0b - PCS[3:2] are configured as PCS feature<br>1b - PCS[3:2] are configured as data bus in 4-bit mode |
| **26**<br>**OUT_CFG** | RW | Output Configuration<br>0b - Output data retains last value when PCS is negated<br>1b - Output data is tristated when PCS is negated. This bit must be set if perform a half duplex transfer. |
| **25-24**<br>**PIN_CFG** | RW | Pin Configuration<br>Configures SIN and SOUT as input/output in 1-bit transfers.<br>00b - SIN is used for input data and SOUT is used for output data. This field must be set to 00b in 2/4-bit mode<br>01b - SIN is used for both input and output data, only half-duplex serial transfers are supported<br>10b - SOUT is used for both input and output data, only half-duplex serial transfers are supported<br>11b - SOUT is used for input data and SIN is used for output data |

| | | |
|---|---|---|
| **18-16**<br>**MAT_CFG** | RW | Match Configuration<br>Configures the RX data match condition.<br>000b - Match is disabled.<br>001b - Reserved<br>010b - Match is enabled. Condition is 1st data word equals MATCH0 OR MATCH1.<br>011b - Match is enabled. Condition is any data word equals MATCH0 OR MATCH1.<br>100b - Match is enabled. Condition is 1st data word equals MATCH0 AND 2nd data word equals MATCH1.<br>101b - Match is enabled. Condition is any data word equals MATCH0 AND the next data word equals MATCH1.<br>110b - Match is enabled. Condition is (1st data word AND MATCH1) equals (MATCH0 AND MATCH1).<br>111b - Match is enabled. Condition is (any data word AND MATCH1) equals (MATCH0 AND MATCH1) |
| **11-8**<br>**PCS_POL** | RW | PCS Polarity<br>Each bit of PCS_POL controls the polarity of one PCS pin. For example, PCS_POL[0] controls PCS0.<br>PCS_POL[n] is 0b - PCSn pin is active low.<br>PCS_POL[n] is 1b - PCSn pin is active high. |
| **2**<br>**INT_PCS** | RW | Internal PCS Enable<br>0b - PCS from chip off<br>1b - PCS generated internally. This field is only used in slave mode. |
| **1**<br>**SCK_LB** | RW | SCK Loopback<br>0b - Input data is sampled on internal SCK edge<br>1b - Input data is sampled on loopback SCK edge. This field is only used in master mode. |
| **0**<br>**MASTER** | RW | Master Mode<br>0b - Configure FCSPI as slave<br>1b - Configure FCSPI as master |

### 57.6.7 FCSPI_DATA_MATCH0 (Data Match Register 0)

Offset : 30h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA0 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA0 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSPI_DATA_MATCH0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**DATA0** | RW | Data 0<br>The value is compared with the RX data if set CFGR0[RDMO]. |

### 57.6.8 FCSPI_DATA_MATCH1 (Data Match Register 1)

Offset : 34h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA1 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA1 | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSPI_DATA_MATCH1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**DATA1** | RW | Data 1<br>The value is compared with the RX data if set CFGR0[RDMO]. |

### 57.6.9 FCSPI_CLK_CFG (Clock Configuration Register)

Offset : 40h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SCKPCS | | | | | | | | PCSSCK | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PCSPCS | | | | | | | | SCKDIV | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSPI_CLK_CFG field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**SCKPCS** | RW | SCK to PCS Delay<br>Configure the delay cycles from the last SCK edge to the PCS negation.<br>DELAY = (SCKPCS + 1)*Tf (Tf is FCSPI clock generated after PRESCALE) |
| **23-16**<br>**PCSSCK** | RW | PCS to SCK Delay<br>Configure the delay cycles from the PCS assertion to the first SCK edge.<br>DELAY = (PCSSCK + 1)*Tf (Tf is FCSPI clock generated after PRESCALE) |
| **15-8**<br>**PCSPCS** | RW | PCS to PCS Delay<br>Configures the delay cycles from the PCS negation to the next PCS assertion.<br>DELAY = (PCSPCS + 2)*Tf (Tf is FCSPI clock generated after PRESCALE) |
| **7-0**<br>**SCKDIV** | RW | SCK Divider<br>Configures the divide ratio of the SCK from the clock generated after PRESCALE.<br>Ratio = SCKDIV + 2 |

### 57.6.10 FCSPI_FIFO_WTM (FIFO Water Mark Register)

Software can set the watermark of the FIFO via writing this register.

Offset : 58h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | | | | | | | | | | | | | RXWATER | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | | | | | | | | | | | | | TXWATER | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSPI_FIFO_WTM field descriptions**

| Field | Type | Description |
|---|---|---|
| **18-16 RXWATER** | RW | Receive FIFO Watermark<br>The RX FIFO flag is set whenever the number of words in the receive FIFO is greater than RXWATER. Cannot write a value equal to or greater than the FIFO size. |
| **2-0 TXWATER** | RW | Transmit FIFO Watermark<br>The TX FIFO flag is set whenever the number of words in the transmit FIFO is equal to or less than TXWATER. Cannot write a value equal to or greater than the FIFO size. |

### 57.6.11 FCSPI_FIFO_STATUS (FIFO Status Register)

Software can get the FIFO status via reading this register.

Offset : 5Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | RXCNT | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | TXCNT | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSPI_FIFO_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16 RXCNT** | RO | Receive FIFO Count<br>Indicates the number of words stored in the RX FIFO. |
| **3-0 TXCNT** | RO | Transmit FIFO Count<br>Indicates the number of words currently stored in the transmit FIFO. |

### 57.6.12 FCSPI_TR_CTRL (Transmit and Receive Control Register)

Software can configure the protocol of the transfer via writing this register.

Offset : 60h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SCK_ | SCK_ | PRESCALE | | | 0 | PCS | | LSBF | BYSW | CT_E | CT_G | RX_M | TX_M | WIDTH | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | POL | PHA | | | | | | | | | N | O | SK | SK | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | | | | FRM_SZ | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

**FCSPI_TR_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**SCK_POL** | RW | SCK Polarity<br>0b - SCK is low when inactive<br>1b - SCK is high when inactive |
| **30**<br>**SCK_PHA** | RW | SCK Phase<br>If SCK_POL = 0, the leading edge means pos-edge of SCK, and the following edge means neg-edge of SCK.<br>If SCK_POL = 1, the leading edge means neg-edge of SCK, and the following edge means pos-edge of SCK.<br>0b - Data is captured on the leading edge of SCK and changed on the following edge of SCK<br>1b - Data is changed on the leading edge of SCK and captured on the following edge of SCK |
| **29-27**<br>**PRESCALE** | RW | Prescaler Value<br>Divide the FCSPI input clock to generate the functional clock.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |
| **25-24**<br>**PCS** | RW | PCS Select<br>00b - Transfer using PCS[0]<br>01b - Transfer using PCS[1]<br>10b - Transfer using PCS[2]<br>11b - Transfer using PCS[3] |

| | | |
|---|---|---|
| **23**<br>**LSBF** | RW | LSB First<br>0b - Data is transferred MSB mode<br>1b - Data is transferred LSB mode |
| **22**<br>**BYSW** | RW | Byte Swap<br>Byte swap will swap the contents of [31:24] with [7:0] and [23:16] with [15:8] for each TX word read from the FIFO and for each RX word stored to the FIFO(or compared with match registers).<br>0b - Disable byte swap<br>1b - Enable byte swap |
| **21**<br>**CT_EN** | RW | Continuous Transfer Mode Enable<br>In master mode (and CT_EN=1), PCS will keep asserted at the end of the frame size, until a TR_CTRL configuration is written that starts a new frame. In slave mode (and CT_EN=1), the FCSPI will only transmit the first FRAMESZ bits, after which the FCSPI will transmit received data until the next PCS negation.<br>0b - Disable continuous transfer mode<br>1b - Enable continuous transfer mode |
| **20**<br>**CT_GO** | RW | Continuous Transfer Go<br>In master mode, CT_GO allows the TR_CTRL configuration to be changed within a continuous transfer. Please ensure that the TR_CTRL configuration is written on a frame size boundary.<br>0b - Start a new transfer (PCS will negate first, then assert again)<br>1b - Continuous transfer with new TR_CTRL configuration (PCS keeps asserted) |
| **19**<br>**RX_MSK** | RW | RX Data Mask<br>When set, the receive data will be masked (RX data is not stored in FIFO).<br>0b - Normal transfer<br>1b - Receive data is masked |
| **18**<br>**TX_MSK** | RW | TX Data Mask<br>When set, the transmit data will be masked (no data loaded from FIFO and output pin is tristated). If the master wants to start a new transfer to receive data from the slave only (without transmit data), then set this field. The TX_MSK bit will be cleared after the transfer completed.<br>0b - Normal transfer<br>1b - Transmit data is masked |

| | | |
|---|---|---|
| **17-16**<br>**WIDTH** | RW | Transfer Width<br>Configures the data bus width of FCSPI. For half-duplex parallel transfers, either RX_MSK or TX_MSK must be set.<br>For 4-bit transfer, if the frame size is 8, the SCK_DIV should be configured equal to or larger than 2.<br>00b - 1 bit transfer<br>01b - 2 bit transfer<br>10b - 4 bit transfer<br>11b - Reserved |
| **11-0**<br>**FRM_SZ** | RW | Frame Size<br>SIZE = FRAMESZ + 1<br>The frame size has the below constraints:<br>1) The frame size must be equal to or larger than 8 bits.<br>2) If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32 bits, each word is loaded from the TX FIFO and stored in the RX FIFO separately.<br>3) If the size of the frame is not divisible by 32, then the last word load from the TX FIFO and stored in RX FIFO will contain the remaining bits. For example, a 40-bit transfer will consist of 2 words: the 1st word is 32 bits, and the 2nd word is 8 bits.<br>4) The minimum word size is 2 bits, that is frame size of 33 bits (or similar) is not supported. |

### 57.6.13 FCSPI_TX_DATA (Transmit Data Register)

The data written to this register will be transmitted.

Offset : 64h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | DATA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | DATA | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSPI_TX_DATA field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-0**<br>**DATA** | WO | TX FIFO Data<br>The written data will be stored in the TX FIFO sequentially. If the written data is 8-bit or 16-bit, then the data will be zero-extended to 32 bits. |

### 57.6.14 FCSPI_RX_STATUS (Receive Status Register)

Offset : 70h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | RX_EMPTY | FD |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**FCSPI_RX_STATUS field descriptions**

| Field | Type | Description |
|---|---|---|
| **1**<br>**RX_EMPTY** | RO | RX FIFO Empty<br>0b - RX FIFO is not empty<br>1b - RX FIFO is empty |
| **0**<br>**FD** | RO | First Data<br>0b - Subsequent data received after PCS assertion<br>1b - First data is received after PCS assertion |

### 57.6.15 FCSPI_RX_DATA (Receive Data Register)

Software can get the received data via reading this register.

Offset : 74h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RX_DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RX_DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCSPI_RX_DATA field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**RX_DATA** | RO | Receive Data<br>Reading this register will get the received data from RX FIFO. |

Flagchip

# Chapter 58 FC Inter-Integrated Circuit (FCIIC)

## 58.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

This chapter describes the FC Inter-Integrated Circuit (FCIIC) Module. It contains the following sections:

- Chip-specific Information
- Introduction
- Features
- Functional Description
- Register Definition

## 58.2 Chip-specific Information

The FC7240F2M contains two FCIIC instances with the configuration below:

- FCIIC master mode operation can support up to 400 kHz in fast mode (400 kbps data rate; effective data rate reduces according to IIC protocol).
- FCIIC slave mode operation can support up to 3.4 MHz in high-speed mode (3.4 Mbps data rate; effective data rate reduces according to IIC protocol).
- The FCIIC module supports System Management Bus (SMBus) and Direct Memory Access (DMA) operation.
- 8 words FCIIC RX FIFO with 8-bit width and 8 words TX FIFO with 11-bit width (8-bit data + 3-bit command).

## 58.3 Overview

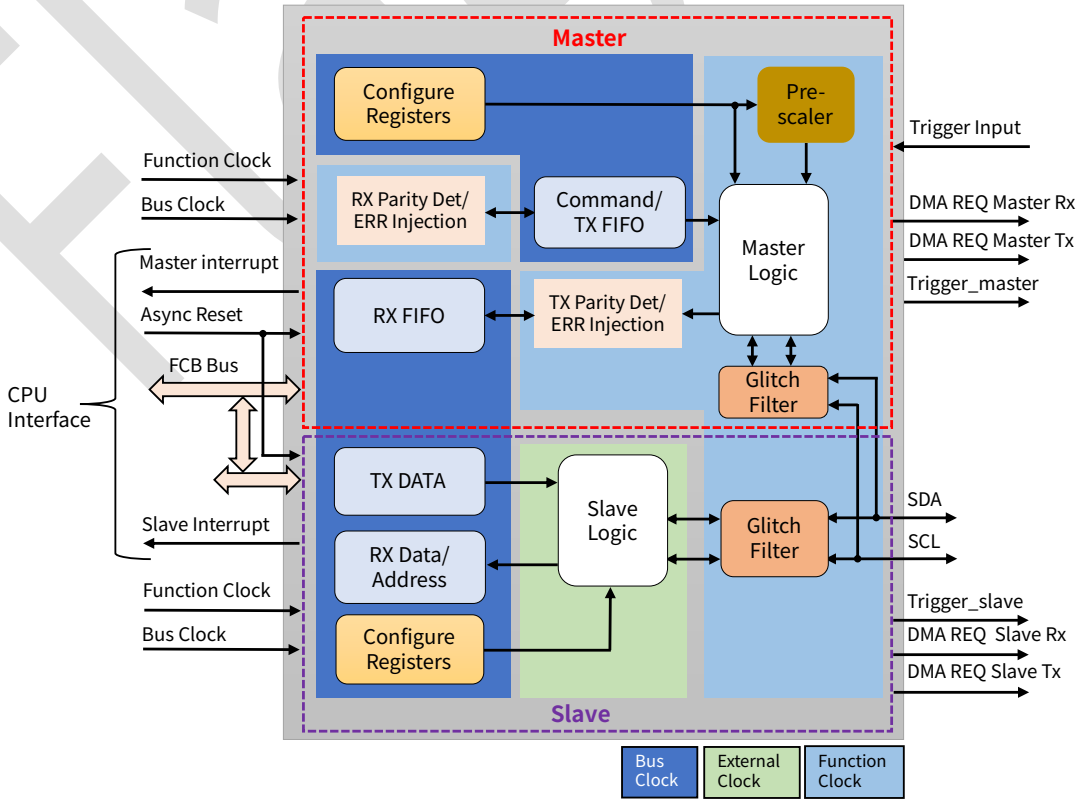
The FCIIC is an Inter-Integrated Circuit (IIC) module that is compatible with IIC-bus specifications and provides an efficient interface to an IIC bus as a master and/or as a slave.

It uses only two bus lines (a serial data line (SDA) and a serial clock line (SCL)) to connect the devices that support the IIC-Bus protocol and can support different transfer speeds in standard mode, fast mode, and high-speed mode operation. It can use little CPU resources with DMA offloading of FIFO register access.

## 58.4 Block Diagram

The following is the block diagram of the FCIIC module.

**Figure 138. FCIIC block diagram**



### 58.4.1 Modes of Operation

The FCIIC supports operation in Run Mode and Debug Mode of the chip:

**Table 85. FCIIC modes of operation**

| Mode | Description |
|---|---|
| Run | Normal operation |
| Wait/Stop/Standby | Not supported |
| Debug | Can continue operating in Debug mode when the Debug Enable bit (FCIIC_MCR[DBGE]) is set. |

## 58.5 Features

The FCIIC supports the following features:

- Compatible with IIC-bus specification.
- Independent logic to support FCIIC operated as a master or slave device.
- Support Standard and Fast modes for different transfer rates, and High-speed mode in FCIIC slave mode.
- Support multi-master with synchronization and arbitration. On the IIC bus, any number of master nodes can be present.
- Support 7-bit and 10-bit bus addressing.
- Support software reset, START byte.
- Flexible clock control by prescaler.
- Clock stretching: Clock stretching pauses a transaction by holding the SCL line LOW. The transaction cannot continue until the line is released HIGH again.
- Several flags and optional interrupts in Master and Slave mode (described in Interrupts and DMA Requests Section).

The FCIIC master mode supports the following features:

- Command/transmit FIFO of 8 words (8-bit transmit data + 3-bit command).
- Receive FIFO of 8 words (8-bit receive data).
- Transmit/Receive FIFO usable as flow controller (seamless DMA flow).
- Command FIFO with the following features to control the transfer:
  - Wait for idle IIC bus before initiating a transfer
  - Initiate (repeated) START and STOP conditions and one or more master-receiver transfers
  - Generate STOP condition (automatically generated when the transmit FIFO is empty)
- Trigger input to control the start time of an IIC bus transfer.
- Flexible receive data match can generate an interrupt on data match and/or discard unwanted data.
- Support configurable bus idle timeout and pin-stuck-low timeout.

The FCIIC slave mode supports the following features:

- Separate IIC slave registers and slave logic to minimize the software overhead between master/slave switching.
- Support for 7-bit or 10-bit bus addressing, address range, SMBus alert, and general call address.
- Optional High-speed mode up to 3.4 Mbit/s.
- Transmit/Receive data register that supports interrupt or DMA requests.
- Software-controllable ACK or NACK, with optional clock stretching on ACK/NACK bit.
- Configurable clock stretching to co-operate with a master device.

## 58.6 Functional Description

### 58.6.1 Bus Speed

According to the IIC-Bus Specification, the transfer mode can be classified as Standard mode, Fast mode, and High-speed mode.

#### *58.6.1.1 Standard Mode*

Standard mode refers to the initial transfer speed mode of the IIC specification which allows up to 100 kbit/s. Standard-mode devices are not upward compatible; they should not be incorporated in a Fast-mode IIC-bus system as they cannot follow the higher transfer rate and unpredictable states would occur.

#### *58.6.1.2 Fast Mode*

Fast-mode devices can receive and transmit at up to 400 kbit/s. The minimum requirement is that they can synchronize with a 400 kbit/s transfer; they can then prolong the LOW period of the SCL signal to slow down the transfer. The protocol, format, logic levels and maximum capacitive load for the SDA and SCL lines are the same as the Standard-mode IIC-bus specification. Fast-mode devices are downward-compatible and can communicate with Standard-mode devices in a 0 to 100 kbit/s IIC-bus system.

#### *58.6.1.3 High-speed Mode*

High-speed mode devices offer a quantum leap in IIC-bus transfer speeds. High-speed mode devices can transfer information at bit rates of up to 3.4 Mbit/s, yet they remain fully downward compatible with Fast-mode or Standard-mode (F/S) devices for bidirectional communication in a mixed-speed bus system. With the exception that arbitration and clock synchronization is not performed during the High-speed mode transfer, the same serial bus protocol and data format are maintained as with the F/S-mode system.

Clock stretching is only allowed after the ACK bit (and before the 1st bit of the next byte). Stretching between bits 2-9 is illegal because the edges of these bits are boosted with an additional current source.

This mode is only implemented in FCIIC Slave mode and needs to be enabled first.

### 58.6.2 General Call, 7-bit and 10-bit Addressing

The general call address is for addressing every device connected to the IIC-bus at the same time using address 0. However, if a device does not need any of the data supplied within the general call structure, it can ignore this address by not issuing an acknowledgment. If a device does require data from a general call address, it acknowledges this address and behaves as a target-receiver.

The IIC address area is restricted to 112 applicable addresses with the 7-bit address mechanism. It turned out, that more combinations were required to prevent problems with the allocation of slave addresses for new devices.

To prevent address clashes, due to the limited range of the 7-bit addresses, a new 10-bit address scheme has been introduced. This enhancement can be mixed with 7-bit addressing and increases the available address range about ten times. After the start condition, a leading "11110" introduces the 10-bit addressing scheme. The last two address bits of the first byte are concatenated with the eight bits of the second byte of the whole 10-bit address. Devices that only use 7-bit addressing simply ignore messages with the leading "11110."

### 58.6.3 Master Mode

Master mode means the FCIIC controls the SCL line and starts up the transmission. The FCIIC master logic operates independently from the slave logic to perform all master mode transfers on the IIC bus.

#### *58.6.3.1 Clock Synchronization and Arbitration*

The IIC bus can only be occupied by one master device at the same time, but more than one device may begin transmitting on a free bus. Clock synchronization and arbitration can be used to decide which device will take control of the bus and complete its transmission.

Clock synchronization is performed using the wired-AND connection of FCIIC interfaces to the SCL line. All devices on the bus will count their LOW period and HIGH period, a synchronized SCL clock is generated with its LOW period determined by the master with the longest clock LOW period, and its HIGH period determined by the one with the shortest clock HIGH period.

During the transfer, the masters constantly monitor SDA and SCL. If one of them detects that SDA is low when it should be high, it assumes that another master is active and immediately stops its transfer.

#### *58.6.3.2 Start, Repeated Start, Stop Condition and Start Byte*

A start condition is the SDA line transitions from high to low while SCL remains high, a stop condition is defined by low to high transition under the same condition of high level on the SCL line. The Repeated Start condition is that the FCIIC continues data sending or receiving without first generating a stop condition.

A Start Byte (0000_0001) is transmitted after the START condition has been transmitted for the MCU detecting an FCIIC transmission when it does not have an on-chip hardware IIC-bus controller.

#### *58.6.3.3 Transmission Commands*

Transmission commands are stored in the transmit FIFO to initiate different FCIIC operations. The operations are defined in FCIIC_MTDR[CMD] register field. Between the START condition and the STOP condition, multiple transmit and receive commands can be inserted, but transmit and receive commands must not be interleaved to comply with the IIC specification. To ensure that only the expected received data is stored in the receive FIFO or compared with the data match logic, the receive data command and the receive data and discard commands can be interleaved.

After completing the receive data command, the FCIIC master will transmit a NACK automatically if the next command in the FIFO is not a receive data command. A NACK is transmitted automatically as well if the transmit FIFO is empty when a receive data command completes and must be followed by a STOP or (repeated) START condition.

The FCIIC master supports 10-bit addressing through a (repeated) START condition, followed by a transmit data byte containing the second address byte, then followed by any number of data bytes with the master-transmit data.

#### *58.6.3.4 Master Mode Operation Flow*

Before Master mode transmission, the following conditions are needed:

- FCIIC is enabled.
- IIC bus needs to be IDLE; the bus state is indicated by FCIIC Bus Busy Flag (FCIIC_MSR[BBF]).
- The transmit FIFO is not empty.
- The trigger input is either asserted or disabled.

Then the FCIIC master initiates a transfer on the IIC bus and involves the following steps:

- Wait the bus idle time equal to ((FCIIC_MCCR[CLKLO] + 1) x FCIIC_MCFGR1[PRESCALE]).
- Transmit a START condition and address byte, the transmission timing parameters need to be configured in Master Clock Configuration 0 (FCIIC_MCCR).
- Perform master-transmit or master-receive transfers configured in FCIIC_MTDR.
- Transmit NACK or ACK on the last byte of a master-receive transfer depending on the next command.
- Transmit a Repeated START or STOP condition to restart a transfer or finish the current transfer.

When clearing the FCIIC_MCR[MEN] to disable the FCIIC master, the FCIIC will continue to empty the transmit FIFO until a STOP condition is transmitted. However, the FCIIC will no longer stall the IIC bus waiting for the transmit or receive FIFO, and after the transmit FIFO is empty, the FCIIC generates a STOP condition automatically.

Under certain conditions, the FCIIC master can stall the IIC bus and pull down the SCL line continuously on the first bit of a byte:

- FCIIC master is enabled and busy with the transmit FIFO being empty and FCIIC_MCFGR1[AUTOSTOP] being clear.
- FCIIC master is enabled and receiving data, but the receive FIFO is full when storing the received data.

#### *58.6.3.5 Received Data Match*

The receive data during master-receiver transfers can be stored in receive FIFO or discarded after data match by different configurations. The receive data match function supports several data match strategies configured in FCIIC_MCFGR1[MATCFG] and FCIIC_MCFGR0[RDMO]:

- Match received data against one of two bytes or against a masked data byte.
- Compare only the first one or two received data words since the last (repeated) START condition.
- Discard all receive data until a data match is detected, using the FCIIC_MCFGR0[RDMO] control bit; to disable this function, clear FCIIC_MCFGR0[RDMO] before clearing FCIIC_MSR[DMF].

Receive data that is already discarded due to the command word cannot cause the data match to set and delay the match on the first received data word until the matched data is received.

#### *58.6.3.6 Pin Configuration*

Open-drain mode: The FCIIC master works in open-drain mode of the SDA and SCL pins by default.

Push-pull mode: In this mode, the SDA pin is tristate for master-receive data bits and master-transmit ACK/NACK bits. Otherwise, the SDA pin is configured as push-pull. To avoid the risk of contention when SDA is push-pull, the pin can be configured for open-drain operation.

### 58.6.4 Slave Mode

Slave mode means the external device controls the SCL line and generates the SCL signal and FCIIC acts as target-device. The FCIIC has independent slave logic to perform all slave mode transfers on the IIC bus.

#### *58.6.4.1 SMBus Alert Address*

The System Management Bus (SMBus) derivates from the IIC bus. It is a single-ended simple two-wire bus, which is typically used for low bandwidth communications. SMBus has an alert response address (ARA), so when the slave device generates an interrupt, it will not immediately clear the interrupt, but keep it until it receives an ARA containing its address from the master device.

When IIC acts as a slave device, it can detect an address match with the SMBus Alert address and then generates a flag.

#### *58.6.4.2 Address Matching*

The FCIIC slave can support 7-bit or 10-bit bus addressing, SMBus alert, and general call address. By configuring the FCIIC_SCFGR1[ADDRCFG], the different addresses will be matched:

- Match ADDR0 in FCIIC_SAMR register in either 7-bit or 10-bit addressing mode.
- Match one address of ADDR0 and ADDR1 in either 7-bit or 10-bit addressing mode.
- Match an address range from ADDR0 to ADDR1 in either 7-bit or 10-bit addressing mode.

By configuring the FCIIC_SCFGR1[GCEN], the General Call Addressing is enabled; by configuring FCIIC_SCFGR1[SAEN], the SMBus Alert Addressing is enabled.

The FCIIC slave can also be configured to detect the high-speed mode master code, and to disable the digital filters and output valid delay time until the next STOP condition is detected.

After a valid address is matched, the FCIIC slave automatically performs slave-transmit or slave-receive transfers until a NACK is detected (unless FCIIC_MCFGR1[IGNACK] is set) or a bit error is detected (the FCIIC slave is driving SDA, but a different value is sampled), or a (repeated) START or STOP condition is detected.

#### 58.6.4.3 Transmit and Receive Data

In FCIIC slave logic, there are separated Transmit Data register and Receive Data register. Both are double-buffered and only updated during a slave-transmit and slave-receive transfer respectively. The Transmit Data register can be configured to only request data after a slave-transmit transfer is detected, or to request new data whenever the Transmit Data register is empty. The Transmit Data register should only be written when the Transmit Data flag is set. The Receive Data register should only be read when the Received Data flag is set, or the Address Valid flag is set and FCIIC_SCFGR1[RXCFG] = 1.

The slave address that was received can be configured to be read from either the Receive Data register or from the Address Status register which should only be read when the Address Valid flag is set.

#### 58.6.4.4 Clock Stretching

Sometimes multiple IIC nodes may be driving the lines at the same time. If any IIC node is driving a line low, then that line will be low. IIC nodes that are starting to transmit a logical one (by letting the line float high) can detect that the line is low, and thereby know that another IIC node is active at the same time.

When node detection is used on the SCL line, it is called clock stretching, and clock stretching is used as an IIC flow control mechanism.

The FCIIC slave supports many configurable options when clock stretching is performed:

- Stretch during the 9th clock pulse of the address byte and the Address Valid flag is set.
- Stretch during the 9th clock pulse of a slave-transmit transfer and the Transmit Data flag is set.
- Stretch during the 9th clock pulse of a slave-receive transfer and the Receive Data flag is set.
- Stretch during the 8th clock pulse of an address byte or a slave-receive transfer and the Transmit ACK flag is set. In high-speed mode, this is disabled.
- To allow additional setup time to sample the SDA pin externally, clock stretching can also be extended for CLKHOLD cycles. In high-speed mode, this is disabled.

Clock stretching can extend for one peripheral bus clock cycle after SDA updates when clock stretching is enabled except the extension by the CLKHOLD configuration. If the peripheral bus clock is so fast, and the first SDA data bit is 1, the SDA setup time requirement may not be met before SCL releasing.

### 58.6.5 Clock and Reset

The clock of FCIIC can be classified as follows:

- FCIIC Functional clock: The functional clock is used by FCIIC for internal master and slave function logic and is asynchronous to the bus clock. It is also used to support digital filter and data hold time configurations by the FCIIC slave. The FCIIC master divides the functional clock by a prescaler and the resulting frequency must be at least 8 times faster than the IIC bus bandwidth.
- External clock: The FCIIC slave logic is clocked directly from the external pins SCL and SDA. This allows the FCIIC slave to remain operational, even when the FCIIC functional clock is disabled.
- Bus clock: The bus clock is only used for bus access to the control and configuration registers. The bus clock frequency must be sufficient to support the data bandwidth requirements of the FCIIC master and slave registers.

The reset of FCIIC can be classified as follows:

- System reset: The logic and registers for the FCIIC master and slave are reset to their default state on a system reset.
- Software reset: Both the FCIIC master and slave logic implement a software reset bit in their Control Register (FCIIC_MCR[RST] and FCIIC_SCR[RST]). The bit will reset master/slave logic and registers to their default state separately, except for the Control Register itself.

### 58.6.6 Timing Parameters

As required in the IIC timing specification, the FCIIC master timing parameters in FCIIC functional clock cycles must be configured as follows:

- tSCL -- SCL clock period: (CLKHI + CLKLO + 2 + SCL_LATENCY) × (2 ^ PRESCALE) × function clock period.
- tHD: STA -- Hold time (repeated) START condition: (SETHOLD +1) × (2 ^ PRESCALE) × function clock period.
- tLow -- LOW period of the SCL clock: (CLKLO + 1) × (2 ^ PRESCALE) × function clock period.
- tHIGH – HIGH period of the SCL clock: (CLKHI + 1 + SCL_LATENCY) × (2 ^ PRESCALE) × function clock period.
- tSU:STA, tSU:STO -- Setup time for a repeated START condition or STOP condition: (SETHOLD + 1 + SCL_LATENCY) × (2 ^ PRESCALE) × function clock period.
- tHD:DAT -- Data Hold Time: (DATAVD + 1) × (2 ^ PRESCALE) × function clock period.
- tSU:DAT -- Data Setup Time: (SDA_LATENCY + 1) × (2 ^ PRESCALE) × function clock period.
- tBUF -- Bus Free Time between a STOP and START condition: (CLKLO + 1 + SDA_LATENCY) × (2 ^ PRESCALE) × function clock period.
- tVD:DAT, tVD:ACK -- Data Valid Time, Data Valid Acknowledge time: (DATAVD + 1) × (2 ^ PRESCALE) × function clock period.

The latency parameters are defined below:

- SCL_LATENCY is defined as ROUNDDOWN ((2 + FILTSCL + SCL_RISETIME) / (2 ^ PRESCALE)).
- SDA_LATENCY is defined as ROUNDDOWN ((2 + FILTSDA + SDA_RISETIME) / (2 ^ PRESCALE)).

These parameters assume the rise time is less than one FCIIC functional clock cycle. The rise time depends on several factors, including the I/O propagation delay, the IIC bus loading, and the external pull-up resistor sizing. A larger rise time increases the number of cycles that the signal takes to propagate through the synchronizer (and glitch filter), which increases the latency.

The following timing restrictions must be enforced to avoid unexpected START or STOP conditions on the IIC bus, or to avoid unexpected START or STOP conditions detected by the FCIIC master. The timing restrictions can be summarized as SDA cannot change when SCL is high outside of a transmitted (repeated) START or STOP condition.

- CLKLO: Minimum value is 0x3; CLKLO x (2 ^ PRESCALE) > SCL_LATENCY.
- CLKHI: Minimum value is 0x1; configure CLKHI to meet the duty cycle requirements in the IIC specification.
- SETUPHOLD: Minimum value is 0x2; SETHOLD × (2 ^ PRESCALE) > SDA_LATENCY.
- DATAVD: 0x1 ~ (CLKLO - SDA_LATENCY – 1); configure DATAVD to meet the data hold requirement in the IIC specification.
- FILTSCL: 0X0 ~ ([CLKLO x (2 ^ PRESCALE)] – 3); FILTSCL and FILTSDA are the only parameters not multiplied by (2 ^ PRESCALE), and the filter clock source is the FCIIC function clock.
- FILTSDA: FILTSCL ~ ([CLKLO x (2 ^ PRESCALE)] – 3); configuring FILTSDA greater than FILTSCL can delay the SDA input to compensate for board-level skew.
- BUSIDLE: Minimum value is (CLKLO+SETHOLD+2) x2; must also be greater than (CLKHI+1).

The FCIIC slave can configure the following timing parameters. These parameters are disabled when FCIIC_SCR[FILTEN] is clear, and when the FCIIC slave detects high-speed mode. When disabled, the FCIIC slave is clocked directly from the IIC bus and may not satisfy all timing requirements of the IIC specification (such as SDA minimum hold time in Standard/Fast mode).

- SDA data valid time from SCL negation to SDA update.
- SCL hold time when clock stretching is enabled to increase setup time when sampling SDA externally.
- SCL glitch filter time.
- SDA glitch filter time.

The FCIIC slave imposes the following restrictions on the timing parameters.

- FILTSDA must be configured to be greater than or equal to FILTSCL (unless compensating for board-level skew between SDA and SCL).
- DATAVD must be configured less than the minimum SCL low period.

### 58.6.7 Interrupts and DMA Requests

In FCIIC master, some sources can generate FCIIC master interrupts and FCIIC master transmit/receive DMA requests. The following sources can generate both master interrupts and DMA requests:

- Transmit Data Flag:
  Generate DMA TX Request. Data can be written to Transmit FIFO, as configured by the Transmit FIFO Watermark MFCR[TXWATER].
- Receive Data Flag:
  Generate DMA RX Request. Data can be read from the Receive FIFO, as configured by the Receive FIFO Watermark MFCR[RXWATER].

The following sources can only generate master interrupts:

- End Packet Flag:
  Master has transmitted a Repeated START or STOP condition.
- STOP Detect Flag:
  Master has transmitted a STOP condition.
- NACK Detect Flag:
  a) During an address byte, the master expected an ACK but detected a NACK.
  b) During an address byte, the master expected a NACK but detected an ACK.
  c) During a master-transmitter data byte, the master detected a NACK.
- Arbitration Lost Flag:
  a) The master lost arbitration due to a START/STOP condition detected at the wrong time;
  b) Or the master was transmitting data but received different data than the data that was transmitted.
- FIFO Error Flag:
  The master is expecting a START condition in the Command FIFO, but the next entry in the Command FIFO is not a START condition.
- Pin Low Timeout Flag:
  Pin low timeout is enabled and SCL (or SDA if configured) is low for longer than the configured timeout.
- Data Match Flag:
  The received data matches the configured data match, but the received data is not discarded due to a command FIFO entry.

The following sources can only generate flag instead of interrupt:

- Master Busy Flag:
  FCIIC master is busy transmitting/receiving data.
- Bus Busy Flag:
  FCIIC master is enabled, and activity is detected on the IIC bus, but a STOP condition has not been detected and a bus idle timeout (if enabled) has not occurred.

In FCIIC slave, some sources can generate FCIIC slave interrupts and FCIIC slave transmit/receive DMA requests. The following sources can generate both slave interrupts and DMA requests:

- Transmit Data Flag:
  Generate a DMA TX Request. Data can be written to the Slave Transmit Data (STDR).
- Receive Data Flag:

Flagchip

Generate a DMA RX Request. Data can be read from the Slave Receive Data (SRDR).

- Address Valid Flag:
  Generate a DMA RX Request. The address can be read from the Slave Address Status (SASR).

The following sources can only generate slave interrupts:

- SMBus Alert Response Flag:
  Slave detected an address match with the SMBus Alert address.
- General Call Flag:
  Slave detected an address match with the General Call address.
- Address Match 1 Flag:
  Slave detected an address match with SAMR[ADDR1] field or using an address range.
- Address Match 0 Flag:
  Slave detected an address match with SAMR[ADDR0] field.
- Transmit or Receive Error Flag:
  a) Transmit data underrun.
  b) Receive data overflow.
  c) Address status overflow (when Receive Data Configuration FCIIC_SCFGR1[RXCFG] = 1). FEF flag can only be set when clock stretching is disabled.
- Bit Error Flag
  Slave was transmitting data, but received different data than what was transmitted.
- STOP Detect Flag
  Slave has detected an address match followed by a STOP condition.
- Repeated Start Flag
  Slave has detected an address match followed by a Repeated START condition.
- Transmit ACK Flag:
  ACK/NACK can be written to the Slave Transmit ACK (STAR).

The following sources can only generate flag instead of interrupt:

- Bus Busy Flag:
  FCIIC slave is enabled, and activity is detected on the IIC bus, but a STOP condition has not been detected.
- Slave Busy Flag:
  FCIIC slave is busy transmitting/receiving data.

### 58.6.8 Peripheral Triggers

The FCIIC can output trigger to other modules in Master mode and Slave mode:

- Master Output Trigger: The FCIIC master generates an output trigger that can be connected to other peripherals on the device. The master output trigger asserts on both a Repeated START or STOP condition, and the master output trigger remains asserted for one cycle of the FCIIC functional clock divided by the prescaler.
- Slave Output Trigger: The FCIIC slave generates an output trigger that can be connected to other peripherals on the device. The slave output trigger asserts on both a Repeated START or STOP condition that occurs after a slave address match, and the slave output trigger remains asserted until the next slave SCL pin negation.

And other modules can input triggers to start the FCIIC transfer:

- Control the start of an FCIIC bus transfer, the FCIIC input trigger can be selected. The input trigger is synchronized and to be detected, the input trigger must assert for at least 2 cycles of the FCIIC functional clock divided by the PRESCALE configuration. When the FCIIC is busy, the input trigger is ignored.

Flagchip

## 58.7 Register Definition

FCIIC0 base address : 4006_6000h

FCIIC1 base address : 4046_7000h

**Register Overview - FCIIC**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **10h** | FCIIC_MCR (Master Control Register) | 32 | RW | 0000_0000h | page |
| **14h** | FCIIC_MSR (Master Status Register) | 32 | W1C | 0000_0001h | page |
| **18h** | FCIIC_MIER (Master Interrupt Enable Register) | 32 | RW | 0000_0000h | page |
| **1Ch** | FCIIC_MDER (Master DMA Enable Register) | 32 | RW | 0000_0000h | page |
| **20h** | FCIIC_MCFGR0 (Master Configuration Register 0) | 32 | RW | 0000_0000h | page |
| **24h** | FCIIC_MCFGR1 (Master Configuration Register 1) | 32 | RW | 0000_0000h | page |
| **28h** | FCIIC_MCFGR2 (Master Configuration Register 2) | 32 | RW | 0000_0000h | page |
| **2Ch** | FCIIC_MCFGR3 (Master Configuration Register 3) | 32 | RW | 0000_0000h | page |
| **40h** | FCIIC_MDMR (Master Data Match Register) | 32 | RW | 0000_0000h | page |
| **48h** | FCIIC_MCCR (Master Clock Configuration Register) | 32 | RW | 0000_0000h | page |
| **58h** | FCIIC_MFCR (Master FIFO Control Register) | 32 | RW | 0000_0000h | page |
| **5Ch** | FCIIC_MFSR (Master FIFO Status Register) | 32 | RO | 0000_0000h | page |
| **60h** | FCIIC_MTDR (Master Transmit Data Register) | 32 | WO | 0000_0000h | page |
| **70h** | FCIIC_MRDR (Master Receive Data Register) | 32 | RO | 0000_4000h | page |
| **110h** | FCIIC_SCR (Slave Control Register) | 32 | RW | 0000_0000h | page |
| **114h** | FCIIC_SSR (Slave Status Register) | 32 | W1C | 0000_0000h | page |
| **118h** | FCIIC_SIER (Slave Interrupt Enable Register) | 32 | RW | 0000_0000h | page |
| **11Ch** | FCIIC_SDER (Slave DMA Enable Register) | 32 | RW | 0000_0000h | page |
| **124h** | FCIIC_SCFGR1 (Slave Configuration Register 1) | 32 | RW | 0000_0000h | page |
| **128h** | FCIIC_SCFGR2 (Slave Configuration Register 2) | 32 | RW | 0000_0000h | page |
| **140h** | FCIIC_SAMR (Slave Address Match Register) | 32 | RW | 0000_0000h | page |
| **150h** | FCIIC_SASR (Slave Address Status Register) | 32 | RO | 0000_4000h | page |
| **154h** | FCIIC_STAR (Slave Transmit ACK Register) | 32 | RW | 0000_0000h | page |
| **160h** | FCIIC_STDR (Slave Transmit Data Register) | 32 | WO | 0000_0000h | page |
| **170h** | FCIIC_SRDR (Slave Receive Data Register) | 32 | RO | 0000_4000h | page |

### 58.7.1 FCIIC_MCR (Master Control Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | 0 | 0 | 0 | | | | DBGEN | 0 | RST | MEN |
| W | | | | | | | RRF | RTF | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_MCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **9**<br>**RRF** | WO | Reset Receive FIFO<br>0b - No effect.<br>1b - Reset the receive FIFO |
| **8**<br>**RTF** | WO | Reset Transmit FIFO<br>0b - No effect.<br>1b - Reset the transmit FIFO |
| **3**<br>**DBGEN** | RW | Debug Enable<br>Enable the Master in Debug mode.<br>0b - Master is disabled in debug mode<br>1b - Master is enabled in debug mode |
| **1**<br>**RST** | RW | Software Reset<br>Software resets all internal master logic and registers except the Master Control (MCR) register. The reset takes effect immediately and remains asserted until negated by software. After set, this bit can be cleared at any time by software.<br>0b - Master logic is not reset<br>1b - Master logic is reset<br>Need to disable the functional clock in PCC before asserting this bit. |
| **0**<br>**MEN** | RW | Master Enable<br>Enable FCIIC Master mode.<br>0b - Master logic is disabled<br>1b - Master logic is enabled |

### 58.7.2 FCIIC_MSR (Master Status Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | BBF | MBF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DMF | PLTF | FEF | ALF | NDF | SDF | EPF | 0 | | | | | | RDF | TDF |
| W | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**FCIIC_MSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **25**<br>**BBF** | RO | Bus Busy Flag<br>Indicate that the FCIIC bus is busy.<br>0b - FCIIC Bus is idle<br>1b - FCIIC Bus is busy |
| **24**<br>**MBF** | RO | Master Busy Flag<br>Indicate that the FCIIC Master is busy.<br>0b - FCIIC Master is idle<br>1b - FCIIC Master is busy |
| **14**<br>**DMF** | W1C | Data Match Flag<br>Set when the received data match with the Match Data in register MDMR[MATCH0] and/or MDMR[MATCH1] fields. The match feature is configured in MCFGR1[MATCFG]. Received data discarded due to CMD field does not cause DMF to set.<br>0b - Have not received matching data<br>1b - Have received matching data |

| | | |
|---|---|---|
| **13**<br>**PLTF** | W1C | Pin Low Timeout Flag<br>Set when the SCL and/or SDA input keeps low longer than PINLOW cycles even if the FCIIC master is idle. The Low Timeout can be set in MCFGR3[PINLOW] fields.<br>The pin low condition pin should be resolved by software, and PLTF cannot be cleared if the pin low timeout continues. To initiate a START condition, the PLTF must be cleared firstly.<br>0b - Pin low timeout has not occurred or is disabled<br>1b - Pin low timeout has occurred |
| **12**<br>**FEF** | W1C | FIFO Error Flag<br>Detect an attempt to send or receive data without generating a (repeated) START condition firstly. This can occur if the transmit FIFO underflows when MCFGR1[AUTOSTOP] = 1. When FEF is set, the FCIIC master sends a STOP condition if it is in busy state, and does not initiate a new START condition until FEF has been cleared.<br>0b - No error<br>1b - Master send or receive data without a START condition |
| **11**<br>**ALF** | W1C | Arbitration Lost Flag<br>Set if either of the following conditions exist:<br>- The FCIIC master transmits a logic one but detects a logic zero on the FCIIC bus<br>- The FCIIC master detects a START or STOP condition while the FCIIC master is transmitting data<br>When ALF sets, the FCIIC master releases the FCIIC bus then turns to idle state, and the FCIIC master does not initiate a new START condition until the ALF has been cleared.<br>0b - Master has not lost arbitration<br>1b - Master has lost arbitration |
| **10**<br>**NDF** | W1C | NACK Detect Flag<br>Set if a NACK is detected by the FCIIC master which was not expecting when transmitting an address or data. When set, the master does not initiate a new START condition but transmit a STOP condition until NDF has been cleared. If a NACK is expected for a given address which is configured by the command word, then the NDF sets if a NACK is not generated.<br>When NDF is set, the FCIIC master automatically transmits a STOP condition if (MCFGR1[AUTOSTOP] = 1) or the transmit FIFO is not empty<br>0b - Unexpected NACK was not detected<br>1b - Unexpected NACK was detected |

| | | |
|---|---|---|
| **9**<br>**SDF** | W1C | STOP Detect Flag<br>Set when a STOP condition is generated by the FCIIC master.<br>0b - A STOP condition has not been generated by Master<br>1b - A STOP condition has been generated by Master |
| **8**<br>**EPF** | W1C | End Packet Flag<br>Set when either a repeated START condition or a STOP condition is generated by the FCIIC master. Does not set when the master first generates a START condition.<br>0b - A STOP or Repeated START condition has not been generated by Master<br>1b - A STOP or Repeated START condition has been generated by Master |
| **1**<br>**RDF** | RO | Receive Data Flag<br>RDF sets whenever the number of words in the receive FIFO is greater than RX Watermark defined in MFCR[RXWATER].<br>0b - Receive Data is not ready<br>1b - Receive data is ready |
| **0**<br>**TDF** | RO | Transmit Data Flag<br>TDF sets whenever the number of words in the transmit FIFO is equal to or less than TX Watermark defined in MFCR[TXWATER].<br>0b - Transmit data is not required<br>1b - Transmit data is required |

### 58.7.3 FCIIC_MIER (Master Interrupt Enable Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DMIE | PLTIE | FEIE | ALIE | NDIE | SDIE | EPIE | 0 | | | | | | RDIE | TDIE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_MIER field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **14**<br>**DMIE** | RW | Data Match Interrupt Enable<br>This bit enables interrupt for data match.<br>0b - Disabled<br>1b - Enabled |
| **13**<br>**PLTIE** | RW | Pin Low Timeout Interrupt Enable<br>This bit enables interrupt for pin low timeout.<br>0b - Disabled<br>1b - Enabled |
| **12**<br>**FEIE** | RW | FIFO Error Interrupt Enable<br>This bit enables interrupt for FIFO error.<br>0b - Enabled<br>1b - Disabled |
| **11**<br>**ALIE** | RW | Arbitration Lost Interrupt Enable<br>This bit enables interrupt for arbitration lost.<br>0b - Disabled<br>1b - Enabled |
| **10**<br>**NDIE** | RW | NACK Detect Interrupt Enable<br>This bit enables interrupt for NACK detect.<br>0b - Disabled<br>1b - Enabled |
| **9**<br>**SDIE** | RW | STOP Detect Interrupt Enable<br>This bit enables interrupt for STOP detect.<br>0b - Disabled<br>1b - Enabled |
| **8**<br>**EPIE** | RW | End Packet Interrupt Enable<br>This bit enables interrupt for end packet.<br>0b - Disabled<br>1b - Enabled |
| **1**<br>**RDIE** | RW | Receive Data Interrupt Enable<br>This bit enables interrupt for receive data.<br>0b - Disabled<br>1b - Enabled |
| **0**<br>**TDIE** | RW | Transmit Data Interrupt Enable<br>This bit enables interrupt for transmit data.<br>0b - Disabled<br>1b - Enabled |

### 58.7.4 FCIIC_MDER (Master DMA Enable Register)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

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

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | RDDE | TDDE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_MDER field descriptions**

| Field | Type | Description |
|---|---|---|
| **1**<br>**RDDE** | RW | Receive Data DMA Enable<br>This bit enables DMA request for receive data.<br>0b - RX DMA request is disabled<br>1b - RX DMA request is enabled |
| **0**<br>**TDDE** | RW | Transmit Data DMA Enable<br>This bit enables DMA request for transmit data.<br>0b - TX DMA request is disabled<br>1b - TX DMA request is enabled |

### 58.7.5 FCIIC_MCFGR0 (Master Configuration Register 0)

Offset : 20h

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

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | RDMO | 0 | | | | | | | | TRGEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_MCFGR0 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **9**<br>**RDMO** | RW | Receive Data Match Only<br>When enabled, all received data will be discarded if it does not cause the Data Match Flag ( MSR[DMF]) to set. When disabled, all received data is stored in the receive FIFO. After MSR[DMF] being set, the RDMO configuration is ignored. Clear RDMO before MSR[DMF] to ensure that no receive data is lost.<br>0b - Received data is stored in the receive FIFO<br>1b - Received data is discarded unless the the Data Match Flag (MSR[DMF]) is set |
| **0**<br>**TRGEN** | RW | Trigger Enable<br>When enabled, the FCIIC master initiates a START condition only if the trigger input is asserted and the bus is idle. The trigger will not affect the repeated START.<br>0b - Trigger input is disabled<br>1b - Trigger input is enabled |

### 58.7.6 FCIIC_MCFGR1 (Master Configuration Register 1)

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | PINCFG | 0 | | | | | | MATCFG | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | TIMECFG | IGNACK | AUTOSTOP | 0 | | | | | PRESCALE | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_MCFGR1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **25**<br>**PINCFG** | RW | Pin Configuration<br>Configure the pin mode for FCIIC. The default configuration is open drain mode, it is Bi-directional open drain for master and slave. When set to 1, it is Bi-directional push-pull for master and slave.<br>0b - open drain mode<br>1b - push-pull mode |

| | | |
|---|---|---|
| **18-16**<br>**MATCFG** | RW | Match Configuration<br>Configure different data match strategies that will cause Data Match Flag to set. See programmed MATCH DATA value in Master Data Match (MDMR) register.<br>000b - Data Match is disabled<br>001b - Reserved<br>010b - Data Match is enabled (1st data byte equals to MDMR[MATCH0] or MDMR[MATCH1])<br>011b - Data Match is enabled (any data byte equals to MDMR[MATCH0] or MDMR[MATCH1])<br>100b - Data Match is enabled (1st data byte equals to MDMR[MATCH0] and 2nd data byte equals to MDMR[MATCH1)<br>101b - Data Match is enabled (any data byte equals to MDMR[MATCH0] and next data byte equals to MDMR[MATCH1)<br>110b - Data Match is enabled (1st data byte AND MDMR[MATCH1] equals to MDMR[MATCH0] and MDMR[MATCH1])<br>111b - Data Match is enabled (any data byte AND MDMR[MATCH1] equals to MDMR[MATCH0] and MDMR[MATCH1]) |
| **10**<br>**TIMECFG** | RW | Timeout Configuration<br>TIMECFG configures the timeout settings for the Pin Low Timeout Flag field (MSR[PLTF]).<br>0b - MSR[PLTF] sets if SCL keeps low longer than the configured timeout<br>1b - MSR[PLTF] sets if either SCL or SDA keeps low longer than the configured timeout |
| **9**<br>**IGNACK** | RW | IGNACK<br>When set, FCIIC Master will ignore the received NACK field and assumed it as an ACK.<br>0b - FCIIC Master receives ACK and NACK normally<br>1b - FCIIC Master assumes a received NACK as an ACK |
| **8**<br>**AUTOSTOP** | RW | Automatic STOP Generation<br>When set, the FCIIC Master will generate a STOP condition whenever it is busy and the transmit FIFO is empty. By configuring the transmit FIFO command, the STOP condition can also be generated.<br>0b - No effect<br>1b - STOP condition will be generated whenever the FCIIC master is busy and the transmit FIFO is empty |

| | | |
|---|---|---|
| **2-0**<br>**PRESCALE** | RW | Prescaler<br>Configures the clock prescaler used for all FCIIC master logic. The prescaler will not be used for the digital glitch filters.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |

### 58.7.7 FCIIC_MCFGR2 (Master Configuration Register 2)

Offset : 28h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | FILTSDA | | | | 0 | | | | FILTSCL | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | BUSIDLE | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_MCFGR2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **27-24**<br>**FILTSDA** | RW | Glitch Filter SDA<br>Configure the FCIIC master digital glitch filters for SDA input.<br>- Configure 0 to disable the glitch filter<br>- SDA input glitches equal to or less than FILTSDA cycles are filtered out<br>- The latency through the glitch filter is equal to FILTSDA cycles, and must be less than the minimum SCL low or high period<br>- PRESCALE is not used in glitch filter |

| | | |
|---|---|---|
| **19-16**<br>**FILTSCL** | RW | Glitch Filter SCL<br>Configure the FCIIC master digital glitch filters for SCL input.<br>- Configure 0 to disable the glitch filter<br>- SCL input glitches equal to or less than FILTSCL cycles are filtered out. The FILTSCL cycles are based on the functional clock.<br>- The latency through the glitch filter is equal to FILTSCL cycles, and must be less than the minimum SCL low or high period<br>- PRESCALE is not used in glitch filter |
| **11-0**<br>**BUSIDLE** | RW | Bus Idle Timeout<br>Configure the bus idle timeout value in clock cycles.<br>- If both SCL and SDA keep high state longer than BUSIDLE cycles, then the FCIIC bus is assumed to be idle and the master can generate a START condition<br>- To disable the Bus Idle Timeout, set the Bus Idle Timeout to zero |

### 58.7.8 FCIIC_MCFGR3 (Master Configuration Register 3)

Offset : 2Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | PINLOW | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PINLOW | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_MCFGR3 field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-8**<br>**PINLOW** | RW | Pin Low Timeout<br>Configure the pin low timeout value in clock cycles.<br>- If either SCL or SDA keeps low longer than (PINLOW * 256) cycles, then MSR[PLTF] is set.<br>- To disable Pin Low Timeout feature, set Pin Low Timeout to zero |

### 58.7.9 FCIIC_MDMR (Master Data Match Register)

Offset : 40h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MATCH1 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | MATCH0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_MDMR field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-16**<br>**MATCH1** | RW | Match 1 Value<br>The value will be compared against the received data when receive data match is enabled in MCFGR1[MATCFG]. |
| **7-0**<br>**MATCH0** | RW | Match 0 Value<br>The value will be compared against the received data when receive data match is enabled in MCFGR1[MATCFG]. |

### 58.7.10 FCIIC_MCCR (Master Clock Configuration Register)

Offset : 48h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DATAVD | | | | 0 | | | SETHOLD | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | CLKHI | | | | 0 | | | CLKLO | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_MCCR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **29-24**<br>**DATAVD** | RW | Data Valid Delay<br>Configure the minimum number of cycles (minus one) that will be used as the data hold time for SDA. But the delay must be less than the minimum SCL low period. |
| **21-16**<br>**SETHOLD** | RW | Setup Hold Delay<br>Configure the minimum number of cycles (minus one) that will be used by the master for following conditions:<br>- Hold time for a START<br>- Setup and hold time for a repeated START<br>- Setup time for a STOP<br>The setup time decided by the time it takes to detect a rising edge on the external SCL pin. Ignore any delay from the board due to external loading, this time can be calculated by (2 + FILTSCL) / 2^PRESCALE cycles. |
| **13-8**<br>**CLKHI** | RW | Clock High Period<br>Configure the minimum number of cycles (minus one) that the SCL clock will be driven high by the master. The time of SCL high is extended by the time it takes to detect a rising edge on the external SCL pin. Ignore any delay from board due to external loading, this time can be calculated by (2 + FILTSCL) / 2^PRESCALE cycles. |
| **5-0**<br>**CLKLO** | RW | Clock Low Period<br>Configure the minimum number of cycles (minus one) that the SCL clock will be driven low by the master. The Clock Low Period value is also used for the minimum bus free time between a STOP and a START condition; this is extended by the time it takes to detect a rising edge on the external SCL pin. Ignore any delay from board due to external loading, this time can be calculated by (2 + FILTSCL) / 2^PRESCALE cycles. |

### 58.7.11 FCIIC_MFCR (Master FIFO Control Register)

Offset : 58h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | RXWATER | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | TXWATER | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_MFCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16**<br>**RXWATER** | RW | Receive FIFO Watermark<br>This field is used to configure the RX FIFO Watermark value. The Receive Data Flag (MSR[RDF]) is set whenever the number of words in the receive FIFO is greater than RXWATER. Writing a value equal to or greater than the FIFO size truncates the value. |
| **3-0**<br>**TXWATER** | RW | Transmit FIFO Watermark<br>This field is used to configure the TX FIFO Watermark value. The Transmit Data Flag (MSR[TDF]) is set whenever the number of words in the transmit FIFO is equal or less than TXWATER. Writing a value equal to or greater than the FIFO size truncates the value. |

### 58.7.12 FCIIC_MFSR (Master FIFO Status Register)

Offset : 5Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | RXCOUNT | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | TXCOUNT | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_MFSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16**<br>**RXCOUNT** | RO | Receive FIFO Count<br>This field stores and can return the number of words in the receive FIFO. |
| **3-0**<br>**TXCOUNT** | RO | Transmit FIFO Count<br>This field stores and can return the number of words in the transmit FIFO. |

### 58.7.13 FCIIC_MTDR (Master Transmit Data Register)

Offset : 60h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | | | | | 0 | | | 0 | | | | | | | |
| W | | | | | | CMD | | | DATA | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_MTDR field descriptions**

| Field | Type | Description |
|---|---|---|
| **10-8**<br>**CMD** | WO | Command Data<br>This field is used to configure the FCIIC command:<br>000b - Transmit DATA[7:0]<br>001b - Receive (DATA[7:0] + 1) bytes<br>010b - Generate STOP condition<br>011b - Receive and discard (DATA[7:0] + 1) bytes<br>100b - Generate (repeated) START and transmit address in DATA[7:0]<br>101b - Generate (repeated) START and transmit address in DATA[7:0]. This transfer expects a NACK to be returned.<br>110b - Reserved<br>111b - Reserved |
| **7-0**<br>**DATA** | WO | Transmit Data<br>Configure the 8-bit transmit data that zero-extends the CMD field. |

### 58.7.14 FCIIC_MRDR (Master Receive Data Register)

Offset : 70h

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

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_MRDR field descriptions**

| Field | Type | Description |
|---|---|---|
| **14**<br>**RXEMPTY** | RO | RX Empty<br>RXEMPTY indicates whether the Master receive data FIFO is empty. This bit is read-only.<br>0b - Receive FIFO is not empty<br>1b - Receive FIFO is empty |
| **7-0**<br>**DATA** | RO | Receive Data<br>Store the received data from RX transfer. Receive data can be discarded due to the CMD field, or the master can be configured to discard non-matching data. These bits are read-only. |

### 58.7.15 FCIIC_SCR (Slave Control Register)

Offset : 110h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | FILTEN | 0 | | RST | SEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_SCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **4**<br>**FILTEN** | RW | Filter Enable<br>Enable or disable digital filter and output delay counter for slave mode. FILTEN should only be updated when the FCIIC Slave is disabled.<br>0b - Disable digital filter and output delay counter for slave mode<br>1b - Enable digital filter and output delay counter for slave mode |

| | | |
|---|---|---|
| **1**<br>**RST** | RW | Software Reset<br>Reset FCIIC Slave logic by software. The reset takes effect immediately and remains asserted until negated by software. After set, this bit can be cleared at any time.<br>0b - Not reset Slave logic<br>1b - Reset Slave logic<br>Need to disable the functional clock in PCC before asserting this bit. |
| **0**<br>**SEN** | RW | Slave Enable<br>Enable or disable the FCIIC Slave mode.<br>0b - Disable FCIIC Slave mode<br>1b - Enable FCIIC Slave mode |

### 58.7.16 FCIIC_SSR (Slave Status Register)

Offset : 114h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | BBF | SBF | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SARF | GCF | AM1F | AM0F | TREF | BEF | SDF | RSF | 0 | | | | TAF | AVF | RDF | TDF |
| W | | | | | w1c | w1c | w1c | w1c | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_SSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **25**<br>**BBF** | RO | Bus Busy Flag<br>Set when FCIIC bus is busy. This bit is read only.<br>0b - FCIIC Bus is idle<br>1b - FCIIC Bus is busy |
| **24**<br>**SBF** | RO | Slave Busy Flag<br>Set when an FCIIC slave is busy. This bit is read only.<br>0b - FCIIC Slave is idle<br>1b - FCIIC Slave is busy |

| | | |
|---|---|---|
| **15**<br>**SARF** | RO | SMBus Alert Response Flag<br>Set when FCIIC Slave detects an address match with the SMBus Alert address. SARF is cleared by reading the Slave Address Status (SASR) register. This bit is read only.<br>0b - SMBus Alert Response is disabled or not detected<br>1b - SMBus Alert Response is enabled and detected |
| **14**<br>**GCF** | RO | General Call Flag<br>Set when a slave has detected the General Call Address. This bit is read only.<br>This flag will be cleared by reading the Slave Address Status (SASR) register<br>0b - Slave has not detected the General Call Address<br>1b - Slave has detected the General Call Address |
| **13**<br>**AM1F** | RO | Address Match 1 Flag<br>Indicate whether the received address has matched the ADDR1 field or ADDR0 to ADDR1 range as configured by SCFGR1[ADDRCFG]. This bit is read only.<br>- Address Match 1 Flag is cleared by reading the Slave Address Status (SASR) register<br>0b - Received address does not match ADDR1 or ADDR0/ADDR1 range<br>1b - Received address matches ADDR1 or ADDR0/ADDR1 range |
| **12**<br>**AM0F** | RO | Address Match 0 Flag<br>Indicate whether the received address has matched the ADDR0 field as configured by SCFGR1[ADDRCFG].<br>- AM0F is cleared by reading the Slave Address Status (SASR) register<br>0b - Received address does not match ADDR0<br>1b - Received address matches ADDR0 |
| **11**<br>**TREF** | W1C | Transmit or Receive Error Flag<br>TREF can only be set when clock stretching is disabled. To eliminate the possibility of overflow occurring, enable clock stretching.<br>TREF also sets due to an address overflow when SCFGR1[RXCFG] is set, otherwise an address overflow is not flagged.<br>0b - Not detect transmit register underflow or receive register overflow<br>1b - Detect transmit register underflow or receive register overflow |
| **10**<br>**BEF** | W1C | Bit Error Flag<br>BEF sets if the FCIIC slave transmits a value but does not detect it on the FCIIC bus. The slave ignores the rest of the transfer until the next (repeated) START condition.<br>0b - A bit error has not been detected by Slave<br>1b - A bit error has been detected by Slave |

| | | |
|---|---|---|
| **9**<br>**SDF** | W1C | STOP Detect Flag<br>SDF sets when the FCIIC slave detects a STOP condition and if the FCIIC slave matched the last address byte.<br>0b - A STOP condition has not been detected by Slave<br>1b - A STOP condition has been detected by Slave |
| **8**<br>**RSF** | W1C | Repeated Start Flag<br>RSF sets when the FCIIC slave detects a repeated START condition and if the FCIIC slave matched the last address byte. The RSF does not set when the slave first detects a START condition.<br>0b - A Repeated START condition has not been detected by Slave<br>1b - A Repeated START condition has been detected by Slave |
| **3**<br>**TAF** | RO | Transmit ACK Flag<br>Indicate whether Transmit ACK/NACK is required. TAF is cleared by writing to the Slave Transmit ACK (STAR) register.<br>0b - Transmit ACK/NACK is not required<br>1b - Transmit ACK/NACK is required |
| **2**<br>**AVF** | RO | Address Valid Flag<br>Indicate whether Address Status Register is valid. AVF is cleared by reading the Slave Address Status (SASR) register. When SCFGR1[RXCFG] = 1, SSR[AVF] is also cleared by reading the Slave Receive Data (SRDR) register.<br>0b - Address Status Register is not valid<br>1b - Address Status Register is valid |
| **1**<br>**RDF** | RO | Receive Data Flag<br>Indicate whether Receive data is ready. RDF is cleared by reading the Slave Receive Data (SRDR) register. When SCFGR1[RXCFG] = 1, RDF is not cleared when reading the Slave Receive Data (SRDR) register and if SSR[AVF] is set.<br>0b - Receive data is not ready<br>1b - Receive data is ready |
| **0**<br>**TDF** | RO | Transmit Data Flag<br>Indicate whether Transmit data is requested. TDF is cleared by writing the to the Slave Transmit Data (STDR) register. When SCFGR1[TXCFG] = 0 and if a NACK or a Repeated START or STOP condition is detected, then TDF is also cleared.<br>0b - Transmit data not required<br>1b - Transmit data is required |

### 58.7.17 FCIIC_SIER (Slave Interrupt Enable Register)

Offset : 118h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SARIE | GCIE | AM1IE | AM0IE | TREIE | BEIE | SDIE | RSIE | 0 | | | | TAIE | AVIE | RDIE | TDIE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_SIER field descriptions**

| Field | Type | Description |
|---|---|---|
| **15**<br>**SARIE** | RW | SMBus Alert Response Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **14**<br>**GCIE** | RW | General Call Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **13**<br>**AM1IE** | RW | Address Match 1 Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **12**<br>**AM0IE** | RW | Address Match 0 Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **11**<br>**TREIE** | RW | Transmit or Receive Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **10**<br>**BEIE** | RW | Bit Error Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **9**<br>**SDIE** | RW | STOP Detect Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **8**<br>**RSIE** | RW | Repeated Start Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

| | | |
|---|---|---|
| **3**<br>**TAIE** | RW | Transmit ACK Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **2**<br>**AVIE** | RW | Address Valid Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **1**<br>**RDIE** | RW | Receive Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |
| **0**<br>**TDIE** | RW | Transmit Data Interrupt Enable<br>0b - Disabled<br>1b - Enabled |

### 58.7.18 FCIIC_SDER (Slave DMA Enable Register)

Offset : 11Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | AVDE | RDDE | TDDE |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_SDER field descriptions**

| Field | Type | Description |
|---|---|---|
| **2**<br>**AVDE** | RW | Address Valid DMA Enable<br>Enable or disable the DMA request. To allow the DMA to read the address from the Slave Receive Data (SRDR) register, both Address Valid DMA request and Receive Data DMA request need enabled, and then set SCFGR1[RXCFG] = 1.<br>0b - DMA request is disabled<br>1b - DMA request is enabled |

| | | |
|---|---|---|
| **1**<br>**RDDE** | RW | Receive Data DMA Enable<br>This bit will enable the DMA request for receive data in Slave mode.<br>0b - RX DMA request is disabled<br>1b - RX DMA request is enabled |
| **0**<br>**TDDE** | RW | Transmit Data DMA Enable<br>This bit will enable the DMA request for transmit data in Slave mode.<br>0b - TX DMA request is disabled<br>1b - TX DMA request is enabled |

### 58.7.19 FCIIC_SCFGR1 (Slave Configuration Register 1)

Offset : 124h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | ADDRCFG | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | HSMEN | IGNACK | RXCFG | TXCFG | SAEN | GCEN | 0 | | | | ACKSTALL | TXDSTALL | RXSTALL | ADRSTALL |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_SCFGR1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **18-16**<br>**ADDRCFG** | RW | Address Configuration<br>Configure different address-matching strategies.<br>000b - Address match 0 (7-bit)<br>001b - Address match 0 (10-bit)<br>010b - Address match 0 (7-bit) or Address match 1 (7-bit)<br>011b - Address match 0 (10-bit) or Address match 1 (10-bit)<br>100b - Address match 0 (7-bit) or Address match 1 (10-bit)<br>101b - Address match 0 (10-bit) or Address match 1 (7-bit)<br>110b - From Address match 0 (7-bit) to Address match 1 (7-bit)<br>111b - From Address match 0 (10-bit) to Address match 1 (10-bit) |

| | | |
|---|---|---|
| **13**<br>**HSMEN** | RW | High Speed Mode Enable<br>Enable detection of the High-Speed Mode master code of slave address 0000_1XX, but does not cause an address match on this code. When set and any High-Speed Mode master code is detected, SCR[FILTEN] and SCFGR1[ACKSTALL] bits are ignored until the next STOP condition is detected.<br>0b - Detection of High Speed Mode master code is disabled<br>1b - Detection of High Speed Mode master code is enabled |
| **12**<br>**IGNACK** | RW | Ignore NACK<br>When set, the FCIIC slave continues transfers after a NACK is detected.<br>0b - Slave stops transfer when NACK is detected<br>1b - Slave does not stop transfer when NACK detected |
| **11**<br>**RXCFG** | RW | Receive Data Configuration<br>0b - Reading the Receive Data register will get the received data and clear the Receive Data flag (MSR[RDF]).<br>1b - When the Address Valid flag (SSR[AVF]) is set, reading the Receive Data Register will get the Address Status register and clear the Address Valid flag. Reading the Receive Data Register when the Address Valid flag is clear, will get received data and clear the Receive Data flag (MSR[RDF]). |
| **10**<br>**TXCFG** | RW | Transmit Flag Configuration<br>Before a NACK is detected at the end of a slave-transmit transfer, the transmit data flag will always be asserted. An extra word will be written to the transmit data FIFO because of this.<br>- When unset, the Transmit Data register will be cleared when a slave-transmit transfer is detected. This will lead to the transmit data flag to assert whenever a slave-transmit transfer is detected; in addition, causes the transmit data flag to de-assert at the end of the slave-transmit transfer.<br>- When set, the Transmit Data flag asserts whenever the Transmit Data register is empty, and negates when the Transmit Data register is full. This allows the Transmit Data register to be filled before a slave-transmit transfer is detected, but can cause the Transmit Data register to be written before a NACK is detected on the last byte of a slave transmit transfer.<br>0b - Transmit Data Flag only asserts during a slave-transmit transfer when the Transmit Data register is empty<br>1b - Transmit Data Flag asserts whenever the Transmit Data register is empty |

| | | |
|---|---|---|
| **9**<br>**SAEN** | RW | SMBus Alert Enable<br>Enable a match on an SMBus alert.<br>0b - Disable match on SMBus Alert<br>1b - Enable match on SMBus Alert |
| **8**<br>**GCEN** | RW | General Call Enable<br>Enable a general call address.<br>0b - Disable General Call address<br>1b - Enable General Call address |
| **3**<br>**ACKSTALL** | RW | ACK SCL Stall<br>To allow software to write the Slave Transmit ACK (STAR) register before the ACK or NACK is transmitted, SCL clock stretching can be enabled during slave-transmit address byte(s) and slave-receiver address and data byte(s). Clock stretching will be operated when transmitting the 9th bit, and is not compatible with high speed mode.<br>When set, it is unnecessary to set either RX SCL Stall (SCFGR1[RXSTALL]) or Address SCL Stall (SCFGR1[ADRSTALL]).When ACKSTALL is enabled and there is an address match on the first byte of a 10-bit address, SSR[AVF] sets allowing software to read the Received Address before writing to the Slave Transmit ACK (STAR) register.<br>0b - Disable Clock stretching<br>1b - Enable Clock stretching |
| **2**<br>**TXDSTALL** | RW | TX Data SCL Stall<br>Enable SCL clock stretching when the SSR[TDF] = 1 during a slave-transmit transfer. Clock stretching should only be operated following the 9th bit, and is compatible with high speed mode.<br>0b - Disable Clock stretching<br>1b - Enable Clock stretching |
| **1**<br>**RXSTALL** | RW | RX SCL Stall<br>Enable SCL clock stretching when SSR[RDF] = 1 during a slave-receive transfer. Clock stretching should only be operated following the 9th bit, and is compatible with high speed mode.<br>0b - Disable Clock stretching<br>1b - Enable Clock stretching |
| **0**<br>**ADRSTALL** | RW | Address SCL Stall<br>Enable SCL clock stretching when SSR[AVF] = 1. Clock stretching only should only be operated following the 9th bit, and is compatible with high speed mode.<br>0b - Disable Clock stretching<br>1b - Enable Clock stretching |

### 58.7.20 FCIIC_SCFGR2 (Slave Configuration Register 2)

Offset : 128h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | FILTSDA | | | | 0 | | | | FILTSCL | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | DATAVD | | | | 0 | | | | CLKHOLD | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_SCFGR2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **27-24**<br>**FILTSDA** | RW | Glitch Filter SDA<br>Configure the FCIIC slave digital glitch filters for SDA input.<br>- Configure 0 to disable the glitch filter<br>- SDA input glitches equal to or less than FILTSDA cycles long are filtered out<br>- The latency through the glitch filter is equal to FILTSDA+3 cycles and less than the minimum SCL low or high period<br>- PRESCALE function is not used in the glitch filter, and the glitch filter cycle count is disabled in high speed mode |
| **19-16**<br>**FILTSCL** | RW | Glitch Filter SCL<br>Configure the FCIIC slave digital glitch filters for SCL input.<br>- Configure 0 to disable the glitch filter<br>- SCL input glitches equal to or less than FILTSCL cycles long are filtered out<br>- The latency through the glitch filter is equal to FILTSCL+3 cycles and less than the minimum SCL low or high period<br>- PRESCALE function is not used in the glitch filter, and the glitch filter cycle count is disabled in high speed mode |
| **13-8**<br>**DATAVD** | RW | Data Valid Delay<br>Configure the SDA data valid delay time equal to FILTSCL+DATAVD+3 cycles.<br>- The data valid delay must be less than the minimum SCL low period<br>- PRESCALE function is not used in Data Valid Delay, and the FCIIC slave data valid delay time is disabled in high speed mode |

| | | |
|---|---|---|
| **3-0**<br>**CLKHOLD** | RW | Clock Hold Time<br>Configure the minimum clock hold time for the FCIIC slave, when clock stretching is enabled.<br>- The minimum hold time is equal to CLKHOLD+3 cycles, (FILTEN must be set to 1 to let this clock hold time take effect)<br>- PRESCALE function is not used in the Clock Hold Time, and the FCIIC slave clock hold time is disabled in high-speed mode |

### 58.7.21 FCIIC_SAMR (Slave Address Match Register)

Offset : 140h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ADDR1 | | | | | | | | | | 0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | ADDR0 | | | | | | | | | | 0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_SAMR field descriptions**

| Field | Type | Description |
|---|---|---|
| **26-17**<br>**ADDR1** | RW | Address 1 Value<br>Configure the Address 1 Value that will be compared against the received address to detect the Slave Address.<br>- In 10-bit mode, the first address byte is compared to { 11110, ADDR1[26:25] } and the second address byte is compared to ADDR1[24:17]<br>- In 7-bit mode, the address is compared to ADDR1[23:17] |
| **10-1**<br>**ADDR0** | RW | Address 0 Value<br>Configure the Address 0 Value that will be compared against the received address to detect the Slave Address.<br>- In 10-bit mode, the first address byte is compared to { 11110, ADDR0[10:9] } and the second address byte is compared to ADDR0[8:1]<br>- In 7-bit mode, the address is compared to ADDR0[7:1] |

### 58.7.22 FCIIC_SASR (Slave Address Status Register)

Offset : 150h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | ANV | 0 | | | RADDR | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_SASR field descriptions**

| Field | Type | Description |
|---|---|---|
| **14**<br>**ANV** | RO | Address Not Valid<br>Indicate whether the received address match with the Address value in SAMR<br>0b - Received Address is valid<br>1b - Received Address is not valid |
| **10-0**<br>**RADDR** | RO | Received Address<br>Whenever the AMF is set, the Received Address update can be updated; the AMF is cleared by reading the Slave Address Status register.<br>- In 7-bit mode, the address byte is stored in RADDR[7:0]<br>- In 10-bit mode, the first address byte is { 11110, RADDR[10:9], RADDR[0] } and the second address byte is RADDR[8:1]. The R/W bit is therefore always stored in RADDR[0]<br>- When ACKSTALL is set, if the first address byte matches in 10-bit mode, then the first address byte is stored in RADDR[7:0] so software can read the Received Address before writing the Transmit ACK. The Received Address then updates with the full 10-bit address if the second address byte matches. |

### 58.7.23 FCIIC_STAR (Slave Transmit ACK Register)

Offset : 154h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | TXNACK |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_STAR field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**TXNACK** | RW | Transmit NACK<br>After FCIIC slave receiving each word, it will transmit an ACK(0) or NACK(1), this can be decided by software configuration; Transmit NACK selects which to use: ACK or NACK.<br>- When SCFGR1[ACKSTALL] is set, a Transmit NACK must be written once for each matching address byte and each received word. ACKSTALL must be set, because that stalls the data transfer until software reads the received word (and decides whether to respond with an ACK or NACK).<br>- To configure the default ACK/NACK, Transmit NACK can also be written when FCIIC Slave is disabled or idle.<br>0b - Write a Transmit ACK for each received word<br>1b - Write a Transmit NACK for each received word |

### 58.7.24 FCIIC_STDR (Slave Transmit Data Register)

Offset : 160h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | DATA | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_STDR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **7-0**<br>**DATA** | WO | Transmit Data<br>This register will store the FCIIC slave transmit data from the configuration |

### 58.7.25 FCIIC_SRDR (Slave Receive Data Register)

Offset : 170h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SOF | RXEMPTY | 0 | | | | | | DATA | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCIIC_SRDR field descriptions**

| Field | Type | Description |
|---|---|---|
| **15**<br>**SOF** | RO | Start Of Frame<br>Indicate whether this is the first data word since a (repeated) START or STOP condition.<br>0b - This is not the first data word since a (repeated) START or STOP condition<br>1b - This is the first data word since a (repeated) START or STOP condition |
| **14**<br>**RXEMPTY** | RO | RX Empty<br>Show the empty status of the receive data register.<br>0b - The Receive Data Register is not empty<br>1b - The Receive Data Register is empty |
| **7-0**<br>**DATA** | RO | Receive Data<br>Contain data received by the FCIIC slave. |

# Chapter 59 FC Universal Asynchronous Receiver Transmitter (FCUART)

## 59.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 59.2 Features

Features of the FC Universal Asynchronous Receiver Transmitter (FCUART) module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Independent FIFO structure for transmitting and receiving:
  - Both Transmit and Receive data FIFO depth are 16
  - Watermark for receiving and transmitting requests are separate configurable
  - Option for the receiver to assert request after a configurable number of idle characters if receive FIFO is not empty
- Programmable baud rate and oversampling ratio
- Use the receiver to monitor other FCUART transmitters
- Programmable data length from 7-bit to 10-bit
- Configurable transmit output and receive input polarity
- Multiple interrupt or DMA request source
- Programmable 1-bit or 2-bit stop bits
- Support operation in Stop mode; FCUART0 supports operation in standby mode
- Three receiver wakeup methods including address mark wakeup, idle line wakeup and receive data match
- Automatic address matching to reduce ISR overhead:
  - Address mark matching
  - Idle line address matching
  - Address alternate matching
- Optional 13-bit break character generation / 11-bit break character detection
- Configurable idle length detection supporting 1, 2, 4, 8, 16, 32, 64 or 128 idle characters
- Request to send (RTS) and clear to send (CTS) hardware control flow support

## 59.3 Overview

The FCUART supports full-duplex, asynchronous, NRZ serial communication and comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver use the same baud rate generate but can operate independently. The following describes each of the blocks of the FCUART.

### 59.3.1 Clocking and Resets

**Table 86. Clocks**

| | |
|---|---|
| **FCUART Functional Clock** | The FCUART functional clock is asynchronous to the bus clock and can remain enabled in stop mode to support transmit/receive, and FCUART0 can operate in standby mode. |
| **Bus Clock** | The bus clock is used for accesses to the registers of the FCUART. |

**Table 87. Resets**

| | |
|---|---|
| **System Reset** | Reset the logic and registers of the FCUART transmitter and receiver. |
| **Software Reset** | The RST register can reset the FCUART logic and registers to their default state, except for the RST register itself. |
| **FIFO Reset** | The FIFO[TXFLUSH] and FIFO[RXFLUSH] can flush the data of transmitter and receiver FIFO. |

### 59.3.2 Baud Rate Generation

Both the receiver and transmitter's baud rate are generated by the modulus counter in the baud rate generator. The SBR field in register BAUD will determine the divisor for the FCUART function clock which will generate the baud rate clock. The transmitter and receiver will use a bit clock which is generated by dividing the baud clock with the oversampling rate (BAUD[OVR_SAMP]).

Baud rate clock is generated by division of the FCUART function clock so it may not give an exact target frequency. Since the baud rate generation is a free-running counter that continues whenever the transmitter or receiver is enabled, and the transmitter bit clock continues whenever the transmitter is enabled so synchronization with the FCUART function clock may cause phase shift.

### 59.3.3 Receiver Function Description

The register STAT[RXINV] can be used to invert the receiver input. The receiver is enabled by CTRL[RE] register. The data length can be set from 7 bits to 10 bits, stop bit can be set to 1 or 2. For the remaining of this section assume the FCUART is set to 8-bit data mode.

After all data bits received and stop bits received, the receiver will write the received data into the receiver data buffer and the receive data buffer full (STAT[RDRFF]) flag will be set. If RDRFF is already set, overrun status occurs and STAT[RORF] flag is set. If an overrun happens, the new data will be lost. The program should read out the data from the receive data buffer after the RDRFF is set to avoid receive overrun.

#### *59.3.3.1 Data Sampling Technique*

The FCUART receiver supports an oversampling rate from 4 to 32. The bit clock is divided into 4 to 32 segments from 1 to OVR_SAMP. The receiver will start by sampling the input logic level at the oversampling rate times of the baud rate clock to search for the start bit. When one logic 0 after three consecutive logic 1 is detected, the receiver will sample three more logic levels at (OVR_SAMP/2), (OVR_SAMP/2)+1, and (OVR_SAMP/2)+2 to make sure it is a real start bit, not a noise. The receiver will assume it is synchronized to a received character if at least two of the three samples are 0. The receiver will restart sampling if another falling edge is detected before the receiver is considered synchronized.

The receiver will use this strategy sampling each bit at (OVR_SAMP/2), (OVR_SAMP/2)+1, and (OVR_SAMP/2)+2 to determine the logic level for that bit. If three sample results are not consistent the noise flag (STAT[NF]) asserts. If the FCUART receiver is configured to sample on both edges of the baud clock, the sample points will be doubled (from 1 to OVR_SAMP*2) compared to the sampling not on both edges. And corresponding sample points will be OVR_SAMP, OVR_SAMP+1, and OVR_SAMP+2. For OVR_SAMP set between 4 to 7, both edge sampling must be enabled, and is optional for higher OVR_SAMP value setting.

The FCUART module can synchronize the oversampling counter with the detected falling edge of the received signal and adjust the sampling window, this is called "synchronization." The falling edge is detected when three consecutive logic 1 is prior to logic 1 -> logic 0 transition. After the start bit falling edge is detected, the receiver will continuously monitor the falling edge and reset the counter once the falling edge is detected. If BAUD[RESYNC_DIS] is cleared, the falling edge detection and resynchronization are performed during all data receive time. Otherwise, the falling edge detection and resynchronization are performed only for the start bit.

#### *59.3.3.2 Receiver Wakeup Operation*

Receiver wakeup and receiver address matching is a mechanism that allows the receiver to ignore a message intended for another receiver.

During receiver wakeup, the receiver will evaluate the first character of the message, and as soon as the receiver finds out the message is intended for another receiver, the receiver wakeup control bit (CTRL[RWC]) will be set. If CTRL[RWC] and STAT[RWUID] are both set, the STAT[IDLEF] is inhibited from setting, and this can eliminate the effort to handle the

unnecessary message. At the end of a message or at the beginning of the next message, the receiver will automatically clear the CTRL[RWC] register so it can wake up in time to detect the next message.

During receiver address matching, the receiver will ignore all data that don't meet the address match requirements.

**Table 88. Receiver wakeup options**

| RWC | MATCH0\|MATCH1 | MATCH_CFG | [RSWMS, RWUID] | Receiver Wakeup |
|---|---|---|---|---|
| 0 | 0 | X | X | Normal operation |
| 1 | 0 | 00 | 00 | Receiver wakeup on idle line, IDLE flag not set |
| 1 | 0 | 00 | 01 | Receiver wakeup on idle line, IDLE flag set |
| 1 | 0 | 00 | 10 | Receiver wakeup on address mark |
| 1 | 1 | 11 | 10 | Receiver wakeup on address match |
| 0 | 1 | 00 | X0 | Address mark address match, IDLE flag not set for discarded characters |
| 0 | 1 | 00 | X1 | Address mark address match, IDLE flag set for discarded characters |
| 0 | 1 | 01 | X0 | Idle line address match |
| 0 | 1 | 10 | X0 | Address alternate match, IDLE flag not set for discarded characters |
| 0 | 1 | 10 | X1 | Address alternate match, IDLE flag set for discarded characters |

#### 59.3.3.2.1 Idle-line Wakeup

The receiver is configured for idle-line wakeup when CTRL[RSWMS] is cleared. The CTRL[RWC] will be automatically cleared when the receiver detects full character time idle bit. The full time character ranges from 9 to 13 bit times according to the setting of CTRL[BMSEL], CTRL[7BMS], BAUD[10BIT_MODE] and BAUD[SBNS].

The idle line that wakes up the receiver will not set the STAT[IDLE] flag when CTRL[RWC] is configured to 1 and STAT[RWUID] configured to 0. If STAT[RWUID] is 1, any idle condition will set the STAT[IDLEF] register regardless of the value of register CTRL[RWC].

The idle-line select (CTRL[ITS]) selects the way that will detect an idle line. With CTRL[ITS] cleared idle bit count after the start bit so the logic 1 in data and the stop bit will toward the judgement of idle condition. On the opposite, if CTRL[ITS] is set the idle bit count after the stop bit, and the idle detection will not be affected by the previous message.

#### 59.3.3.2.2 Address-mark Wakeup

The receiver is configured to address-mark wakeup when CTRL[RSWMS] is set. The CTRL[RWC] will be automatically cleared when the receiver detects the MSB of the received data is logic 1 and STAT[RDRFF] will be set. If parity is enabled, the second MSB will be used for address-mark wakeup.

#### 59.3.3.2.3 Data Match Wakeup

The receiver is configured for data match wakeup when CTRL[RWC] is set to 1, CTRL[RSWMS] is set to 1, and BAUD[MATCH_CFG] is set to 2'b11. In this mode, the CTRL[RWC] will be automatically cleared when the received data matches the MATCH[MATCH0] or MATCH[MATCH1] according to the configuration of BAUD[MAEN0] and BAUD[MAEN1].

#### 59.3.3.2.4 Address Match Operation

When BAUD[MAEN0] or BAUD[MAEN1] bit is set and BAUD[MATCH_CFG] set to 2'b00, the address match function is enabled. In this condition, if the MSB of the received data is logic 1 (or the second MSB is logic 1 when parity is enabled), it will be considered as an address and compared with the related MATCH[MATCH0] or MATCH[MATCH1]. If the comparison matches, the data will be transferred to the receive buffer and STAT[RDRFF] is set. All subsequent data received with MSB logic 0 (or second MSB is logic 0 when parity is enabled) are considered to be data associated with the address and are stored in the receive data buffer. If no address match occurs, then the following data will be discarded.

If BAUD[MAEN0] and BAUD[MAEN1] are both set to 1, a marked address is compared with both match registers. And if the address match with either one of the two register data will be stored in the receive data buffer.

#### 59.3.3.2.5 Alternate Match Operation

When both BAUD[MAEN0] and BAUD[MAEN1] are set to 1 and BAUD[MATCH_CFG] is set to 2'b10, the alternate match operation is enabled. After data received matches with the MATCH[MATCH0], it will be stored in the receive data buffer and STAT[RDRFF] is set. And all subsequent data will be considered as data and also will be stored in the receive data buffer until a character received matches with MATCH[MATCH1]. The data that matches MATCH[MATCH1] and all following data will be discarded until data match MATCH[MATCH0] is received.

#### 59.3.3.2.6 Idle Match Operation

When BAUD[MAEN0] or BAUD[MAEN1] bit is set and BAUD[MATCH_CFG] is set to 2'b01, idle match operation is enabled. In this condition, the first data received after the idle line condition is considered as an address and will be compared with the associated MATCH[MATCH0] or MATCH[MATCH1] register. If the comparison is matched, the data will be stored in the receive data buffer, and STAT[RDRFF] will be set. All subsequent data will be considered as data related to the address and will store in the receive data buffer until the next idle line condition is detected. If no address match occurs, then all following data will be discarded until the next idle line condition is detected. If BAUD[MAEN0] and BAUD[MAEN1] are all set to 1, the first data after an idle line is compared with both MATCH[MATCH0] or MATCH[MATCH1] fields and if it matches with either one the data will be stored to the receive data buffer.

#### 59.3.3.2.7 Hardware Flow Control

The receiver can automatically control the RTS. With request-to-send functionality enabled, the receiver automatically negates the RTS if the receiver data buffer is full or a start bit is detected and will cause the receiver data buffer to be full. The receiver asserts RTS when the amount of data in the receive data buffer is not full, and no start bit is detected which will cause the receive data buffer to be full. Notice that even RTS negates the receiver will continuously receive data until an overrun occurs.

### 59.3.4 Transmitter Function Description

Since the CTRL[TXINV] will be cleared after reset, the transmitter output idle state is logic high in default. The transmitter will be enabled by setting the CTRL[TE] field. This will queue a preamble character (full character frame of the idle state). Then the transmitter will remain idle until the transmit data is available in the transmit data buffer by writing data to the DATA register.

The transmit shift register is 9-bit to 13-bits long depending on the setting of the CTRL[BMSEL], CTRL[7BMS], BAUD[10BIT_MODE] and BAUD[SBNS] register fields. This section will take 8-bit data 1 stop bit mode as an example. With this configuration, the transmit shift register contains a start bit, 8 data bits, and a stop bit. When the shift register is available for new data, the value in the transmit data buffer will be loaded into it. And the transmit data register empty (STAT[TDREF]) will be set until new data is written to the transmit data buffer. If no new data is in the transmit data buffer after all bits shift out the TXD pin, the transmitter will set the transmit flag and enter idle mode waiting for more data to transfer.

Writing 0 to CTRL[TE] will not terminate the transfer immediately. The transmitter will finish the current transfer activity, then the transmitter will be disabled, and no more data will be transferred.

#### *59.3.4.1 Functional Description*

##### 59.3.4.1.1 Send Break and Queued Idle

Configuring the CTRL[SBK] bit can send break characters which are a full character of logic 0, 9-bit to 12-bit times including the start and stop bits. Setting STAT[BCGL] can send a longer break of 13-bit times. The recommended program sequence is waiting until STAT[TDREF] asserts first, then writing 1 and then 0 to the CTRL[SBK]. This will queue a break character to be sent when the transmitter is ready. If CTRL[SBK] remains 1 after the queued break is loaded into the shift register, a new break character is queued. When the FCUART receives a break character, the received data is 0 and a frame error will be detected. Similarly, the break character can be sent by setting DATA[FETSC] and writing 0 to the data buffer. This supports transmitting the break character as part of the normal data stream and also allows the DMA to transmit a break character.

When using idle-line wakeup, a full character time of logic 1(idle) should be sent between messages to wake up the sleeping receiver. The recommended program sequence is waiting until STAT[TDREF] asserts first, then writing 0 and then 1 to the CTRL[TE] bit. This will queue an idle character to be sent as soon as the transmitter is ready. The idle character can also be transferred by setting DATA[FETSC] and writing to the data buffer with all bits logic 1. This supports transmitting the idle character as part of the normal data stream and also allows the DMA to transmit an idle character.

As shown in the below table, the setting value of BAUD[SNBS], STAT[BCGL], CTRL[BMSEL], CTRL[7BMS], and BAUD[10BIT_MODE] will influence the break character length.

**Table 89. Break character length**

| BCGL | BMSEL | 10BIT_MODE | 7BMS | SBNS | Break Character Length |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 10 bit times |
| 0 | 0 | 0 | 0 | 1 | 11 bit times |
| 0 | 0 | 0 | 1 | 0 | 9 bit times |
| 0 | 0 | 0 | 1 | 1 | 10 bit times |
| 0 | 1 | 0 | X | 0 | 11 bit times |
| 0 | 1 | 0 | X | 1 | 12 bit times |
| 0 | X | 1 | X | 0 | 12 bit times |
| 0 | X | 1 | X | 1 | 13 bit times |
| 1 | 0 | 0 | 0 | 0 | 13 bit times |
| 1 | 0 | 0 | 0 | 1 | 13 bit times |
| 1 | 0 | 0 | 1 | 0 | 12 bit times |
| 1 | 0 | 0 | 1 | 1 | 12 bit times |
| 1 | 1 | 0 | X | 0 | 14 bit times |
| 1 | 1 | 0 | X | 1 | 14 bit times |
| 1 | X | 1 | X | 0 | 15 bit times |
| 1 | X | 1 | X | 1 | 15 bit times |

##### 59.3.4.1.2 Hardware Flow Control

The transmitter supports hardware flow control by suspending the transmission with the clear-to-send (CTS) pad. If the clear to send function is enabled, the data will be transferred when CTS is asserted. When CTS asserts in the middle of

transmission the TXD will remain in the mark state until CTS reasserts and the remaining data will continue to be transferred. To guarantee a new transmission is started when the transmitter is idle with data to send the CTS must assert for longer than one-bit period.

#### 59.3.4.1.3 Transceiver Driver Enable

The transmitter can use RTS as an enable signal which is used for an external transceiver. With the request-to-send function enabled, RTS asserts one-bit time before sending the start bit if a character is written into an empty transmitter data buffer. RTS will assert if the transmitter data buffer is not empty. RTS will negate one-bit time after all data bits in the transmit data buffer are sent. Sending break character will also assert RTS.

### 59.3.5 Interrupts and Status Flags

FCUART can generate all synchronous interrupts, but part of the interrupts can be generated as asynchronous interrupts. The asynchronous interrupts of FCUART include:

- Transmit complete interrupt (TCF field in STATUS register)
- Transmit data register empty interrupt (TDREF field in STATUS register)
- Receive data register full interrupt (RDRFF field in STATUS register)
- Idle line interrupt (IDLEF field in STATUS register)
- LIN break detect interrupt (LBKDIF in STATUS register)
- RX input active edge interrupt (RPAEIF in STATUS register)
- Receiver overrun interrupt (RORF in STATUS register)

When the chip enters STOP or STANDBY low power mode FCUART instances can only generate asynchronous interrupts. And when the chip is in RUN mode FCUART instances will generate synchronous interrupts.

The transmit data register empty flag (STAT[TDREF]) asserts when there is space in the transmit data buffer for more transmit data written to the DATA register. If transmit interrupt enable CTRL[TIE] is set to 1, an interrupt will request when STAT[TDREF] is set. Transmit complete flag (STAT[TCF]) indicates that the transmit already finished transferring all data, break characters, preamble and now is idle. If transmit complete interrupt enable (CTRL[TCIE]) is set, an interrupt will request when STAT[TCF] is set. Software polling can be used instead of interrupt when the interrupt is disabled.

The receive data register fill flag (STAT[RDRFF]) will be set when there is enough data in the receive data buffer. Reading data from the data register can clear the flag. There is an internal logic that prevents the IDLE status flag from being set repeatedly when RXD remains idle for a period. IDLE can be cleared by writing 1 to STAT[IDLEF], after it is cleared, it cannot become set again until new data is received and STAT[RDRFF] is set.

If an error is detected during the reception of the data, related error flags including the noise flag (STAT[NF]), framing error flag (STAT[FEF]), and parity error flag (STAT[PF]) will be set. If STAT[RDRFF] is set after the data is received, the error flag will be set at the same time as STAT[RDRFF]. The overrun flag (STAT[RORF]) will be set if new data is received and the STAT[RDRFF] already asserts. When an overrun happens, the newly received data will be discarded. With the receiver enabled the RDX pin active edge interrupt flag (STAT[RPAEIF]) will be set if an active edge on the RXD serial input is detected. If the received data matches the value of MATCH[MATCH0] or MATCH[MATCH1] then the STAT[M0F] or STAT[M1F] flags are set.

### 59.3.6 Other FCUART Functions

#### *59.3.6.1 Data Modes*

The FCUART can be configured from 7-bit to 10-bit mode. Setting CTRL[7BMS] to operate in 7-bit mode, setting CTRL[BMSEL] to operate in 9-bit mode, and setting BAUD[10BIT_MODE] to operate in 10-bit mode. For the transmit data buffer when operating in 9-bit or 10-bit mode, the ninth and tenth bits are stored in CTRL[T8] and CTRL[T9]. For receiver data buffer the

ninth and tenth bits are stored in CTRL[R8] and CTRL[R9]. They are also accessible through 16-bit or 32-bit access of the DATA register. When accessing the transmit data buffer by byte, write to CTRL[T8] and CTRL[T9] before writing to DATA[7:0]. For 16-bit or 32-bit access to the DATA register, all transmit bits are accessed at the same time.

When transmitting new data that have the same ninth and tenth bits of the previous data, there is no need to write CTRL[T8] and CTRL[T9] again. The 9-bit data mode is usually used with parity to make eight bits data plus the parity in the ninth bit, or it is used with address-mark wakeup and the ninth data bit can serve as the wakeup bit. The 10-bit data mode is usually used with parity and address-mark wakeup so the ninth data bit can serve as the wakeup bit and the tenth bit as the parity bit. In custom protocols, the ninth or tenth bits can also serve as software-controlled markers.

#### *59.3.6.2 Loop Mode*

When CTRL[LOOPMS] set to 1, the configuration value of CTRL[RXSRC] will determine the FCUART work in loop mode (CTRL[RXSRC] = 0) or single-wire mode (CTRL[RXSRC] = 1). In loop mode, the transmit output signal will loop back to the receive input and the RXD pad will not be used, this mode is sometimes used to check software that is independent of external connection. The CTRL[LOOPMS] and CTRL[RXSRC] cannot be set to 1 at the same time to avoid a glitch of fcuart_rx before configuring the pin option.

#### *59.3.6.3 Single-wire Mode*

When CTRL[LOOPMS] is set to 1, and CTRL[RXSRC] set to 1, the FCUART will work in single-wire mode. Single-wire mode implements a half-duplex serial connection. The receiver will internally connect to transmitter output, and the RXD pin will not be used. In single-wire mode, the user should use CTRL[TXDIR] to control the direction of the TXD. When CTRL[TXDIR] is set to 1, the TDX is output signal connection to the transmitter, and the FCUART now only can transmit data. When CTRL[TXDIR] is set to 0, the TXD is used as an input that connects to the receiver, and the transmitter is temporarily disconnected from the TXD.

#### *59.3.6.4 Monitor Mode*

The FCUART can use its receiver to monitor the transition of the other FCUART. In this mode, the TXD signal of FCUART which will be monitored will connect to the monitor's RXD. The Monitor should be configured to the right state to monitor the other FCUART, such as the baud rate and data mode. See SMISC peripheral mux control (PERIMUXx) register for more details about the register configuration.

### 59.3.7 Peripheral Triggers

For the connection of the FCUART peripheral triggers, refer to the TRGSEL section for details.

#### *59.3.7.1 Input Trigger*

The FCUART supports one peripheral input trigger, it can be configured in one of the following ways by setting register PIN_CFG.

- The input trigger is used instead of the RXD input.
- The input trigger is used instead of the CTS pin input. In this condition, the input trigger must assert for longer than one bit clock period when the transmitter is idle with data to send to guarantee a new transmission.
- The input trigger is used to modulate the transmit output (ANDed with the TXD output).

#### *59.3.7.2 Output Triggers*

The FCUART generates the following output triggers.

- The transmit data trigger asserts at the end of each transmitted data and negates after one bit period.
- The receive data trigger asserts at the end of each received data that is written to the receive data buffer for one oversampling clock period.

- The receive idle trigger asserts when the idle flag is set for one oversampling clock period.

## 59.4 Register Definition

FCUART0 base address : 4006_8000h
FCUART1 base address : 4006_9000h
FCUART2 base address : 4006_A000h
FCUART3 base address : 4006_B000h
FCUART4 base address : 4046_8000h
FCUART5 base address : 4046_9000h
FCUART6 base address : 4046_A000h
FCUART7 base address : 4046_B000h

**Register Overview - FCUART**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **8h** | FCUART_RST (Software Reset Register) | 32 | RW | 0000_0000h | page |
| **Ch** | FCUART_PIN_CFG (Pin Configuration Register) | 32 | RW | 0000_0000h | page |
| **10h** | FCUART_BAUD (Baud Rate Register) | 32 | RW | 0F00_0004h | page |
| **14h** | FCUART_STAT (Status Register) | 32 | RW | 00C0_0000h | page |
| **18h** | FCUART_CTRL (Control Register) | 32 | RW | 0000_0000h | page |
| **1Ch** | FCUART_DATA (Data Register) | 32 | RW | 0000_1000h | page |
| **20h** | FCUART_MATCH (Match Address Register) | 32 | RW | 0000_0000h | page |
| **24h** | FCUART_MODIR (Modem IrDA Register) | 32 | RW | 0000_0000h | page |
| **28h** | FCUART_FIFO (FIFO Register) | 32 | RW | 00C0_0011h | page |
| **2Ch** | FCUART_WATERMARK (Watermark Register) | 32 | RW | 0000_0000h | page |

### 59.4.1 FCUART_RST (Software Reset Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | RST | 0 |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCUART_RST field descriptions**

| Field | Type | Description |
|---|---|---|
| **1**<br>**RST** | RW | Software Reset<br>Resets the internal logic and all other registers. The reset will remain asserted before software clears this bit.<br>0b - Module is not reset.<br>1b - Module is reset. |

### 59.4.2 FCUART_PIN_CFG (Pin Configuration Register)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | TRGSEL | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCUART_PIN_CFG field descriptions**

| Field | Type | Description |
|---|---|---|
| **1-0**<br>**TRGSEL** | RW | Trigger Select<br>Configures the input trigger usage. Only set this field when the transmitter and receiver are disabled.<br>00b - Input trigger disabled.<br>01b - Input trigger used instead of RXD pin input.<br>10b - Input trigger used instead of CTS_B pin input.<br>11b - Input trigger used to modulate the TXD pin output. The TXD pin output (after TXINV configuration) is ANDed with the input trigger. |

### 59.4.3 FCUART_BAUD (Baud Rate Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAEN | MAEN | 10BI | OVR_SAMP | | | | | TDMA | 0 | RDMA | RIDM | MATCH_CFG | | BEDG | RESY |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | 0 | 1 | T_MODE | | | | | EN | | EN | AEN | | | E_SAMP | NC_DIS | |
| RST | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | LBKDIE | RIAEIE | SBNS | SBR | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**FCUART_BAUD field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**MAEN0** | RW | Match Mode Enable 0<br>0b - Match mode disabled<br>1b - Enables automatic address matching or data matching mode for MATCH[MATCH0]. |
| **30**<br>**MAEN1** | RW | Match Mode Enable 1<br>0b - Normal operation.<br>1b - Enables automatic address matching or data matching mode for MATCH[MATCH1]. |
| **29**<br>**10BIT_MODE** | RW | 10-bit Mode Select<br>The 10-bit mode configures the transfer and receive data to be 10 bits. This bit should only be changed when the transmitter and receiver are disabled.<br>0b - Receiver and transmitter data are not 10 bit.<br>1b - Receiver and transmitter data are 10 bit. |
| **28-24**<br>**OVR_SAMP** | RW | Oversampling Ratio<br>This field configures the oversampling ratio for the receiver. The true oversampling ratio will be the written value adds 1. Only set this field when the transmitter and receiver are disabled.<br>00000b - Writing 0 will make oversampling ratio to be 16.<br>00001b - Reserved<br>00010b - Reserved<br>00011b - Oversampling ratio set to 4.<br>00100b - Oversampling ratio set to 5.<br>00101b - Oversampling ratio set to 6.<br>...<br>11111b - Oversampling ratio of 32. |

| | | |
|---|---|---|
| **23**<br>**TDMAEN** | RW | Transmitter DMA Enable<br>TDMAEN makes the transfer generate a DMA request when transmit data register empty flag STAT[TDREF] is set.<br>0b - Transfer DMA request disabled.<br>1b - Transfer DMA request enabled. |
| **21**<br>**RDMAEN** | RW | Receiver Full DMA Enable<br>RDMAEN makes the receiver generate a DMA request when receiver data register full flag STAT[RDRFF] is set.<br>0b - Receiver DMA request disabled.<br>1b - Receiver DMA request enabled. |
| **20**<br>**RIDMAEN** | RW | Receiver Idle DMA Enable<br>RIDMAEN makes the receiver generate a DMA request when receiver idle flag STAT[IDLEF] is set.<br>0b - DMA request disabled.<br>1b - DMA request enabled. |
| **19-18**<br>**MATCH_CFG** | RW | Match Configuration<br>Configures the match addressing mode. Only set this field when the transmitter and receiver are disabled.<br>00b - Address Match Wakeup<br>01b - Idle Match Wakeup<br>10b - Alternate Match<br>11b - Enables RWC on Data Match and Match On/Off for transmitter CTS input |
| **17**<br>**BEDGE_SAMP** | RW | Both Edge Sampling<br>After enabling this bit, the received data will be sampled by baud clock on both rising and falling edges. When OVR_SAMP is set between 3 to 6, this field must be set, but is optional for higher ratios. Only set this field when the receiver is disabled.<br>0b - Receiver both edge sampling disabled.<br>1b - Receiver both edge sampling enabled. |
| **16**<br>**RESYNC_DIS** | RW | Resynchronization Disable<br>When enabled, disables the resynchronization of the received data when falling edge of the received data is detected. Only set this field when the receiver is disabled.<br>0b - Receive data resynchronization enabled.<br>1b - Receive data resynchronization disabled. |
| **15**<br>**LBKDIE** | RW | LIN Break Detect Interrupt Enable<br>Enable this bit when the LIN break detect flag STAT[LBKDIF] is set, then the corresponding interrupt will be generated.<br>0b - LIN Break Detect Interrupt disabled.<br>1b - LIN Break Detect Interrupt enabled. |

| | | |
|---|---|---|
| **14**<br>**RIAEIE** | RW | RX Input Active Edge Interrupt Enable<br>Enabling this bit will generate an interrupt when the receive input active edge STAT[RPAEIF] is set. Notice that when RIAEIE is set and changing the LOOP or RXSRC field in CTRL register will cause the RPAEIF to be set.<br>0b - RX Input Active Edge Interrupt disabled.<br>1b - RX Input Active Edge Interrupt enabled. |
| **13**<br>**SBNS** | RW | Stop Bit Number Select<br>SBNS configures the stop bits number. Change this bit only when the transmitter and receiver are disabled.<br>0b - One stop bit.<br>1b - Two stop bits. |
| **12-0**<br>**SBR** | RW | Baud Rate Modulo Divisor.<br>Configure the modulo divide rate for the baud rate generator. The baud rate calculation equation is "baud clock / ((OVR_SAMP+1) × SBR)". Only set this field when the transmitter and receiver are disabled. |

### 59.4.4 FCUART_STAT (Status Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LBKDIF | RPAEIF | MSBF | RXINV | RWUID | BCGL | LBKDE | RAF | TDREF | TCF | RDRFF | IDLEF | RORF | NF | FEF | PEF |
| W | w1c | w1c | | | | | | | | | | w1c | w1c | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | M0F | M1F | 0 | | | | | | | | | | RPEF | TPEF | 0 | |
| W | w1c | w1c | | | | | | | | | | | w1c | w1c | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCUART_STAT field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**LBKDIF** | W1C | LIN Break Detect Interrupt Flag<br>When LBKDE is enabled and a LIN break character is detected, this field will assert. This register is write one clear (W1C).<br>0b - No LIN break character detected.<br>1b - LIN break character detected. |

| | | |
|---|---|---|
| **30**<br>**RPAEIF** | W1C | RXD Pin Active Edge Interrupt Flag<br>When receiver is enabled and active edge is detected, this field will assert. Notice that the RXINV field will have impact on this field. This register is write one clear (W1C).<br>0b - Active edge on the receive pin not detected.<br>1b - Active edge on the receive pin detected. |
| **29**<br>**MSBF** | RW | MSB First<br>Configure the bit order of the data that will be transferred or received. Only set this bit when the transmitter and receiver are disabled.<br>0b - The first bit that transmitted or received following the start bit is LSB (bit0).<br>1b - The first bit that is transmitted or received following the start bit is MSB (bit7 for example in 8 bit mode). |
| **28**<br>**RXINV** | RW | Receive Data Inversion<br>When RXINV is enabled, the receiver will reverse the input receive signal polarity. Only set this bit when the receiver is disabled.<br>0b - Receive signal not inverted.<br>1b - Receive signal inverted. |
| **27**<br>**RWUID** | RW | Receive Wake Up Idle Detect<br>For address match wakeup, this field controls whether IDLE bit is asserted when there is no address match. For RWC set to 1, this field controls whether the idle character that wakes up the receiver sets the IDLE bit. Only set this bit when the receiver is disabled.<br>0b - For address match wakeup, when there is no address match, the IDLE bit will not be set. For receive standby state (RWC = 1), when an idle character is detected, the IDLE bit will not be set.<br>1b - For address match wakeup, when there is no address match, the IDLE bit will be set. For receive standby state (RWC = 1), when an idle character is detected, the IDLE bit will be set. |
| **26**<br>**BCGL** | RW | Break Character Generation Length<br>BCGL configures the selection of the transmitted break character length. This bit has no effect on the frame error detection. Only set this bit when the transmitter is disabled. Setting to CTRL[SBK] or write the transfer FIFO with DATA[FETSC] set and DATA[RXTXDATA[9]] clear will send a break character.<br>0b - Set break character length to 9 to 13 bit times.<br>1b - Set break character length to 12 to 15 bit times. |

| | | |
|---|---|---|
| **25**<br>**LBKDE** | RW | LIN Break Detection Enable<br>Enabling this bit will select a longer break character detection length and receive data is not stored in the receive data buffer.<br>0b - Normal break character can be detected with LIN break detection disabled.<br>1b - LIN break detection is enabled. (For example, in 8 bit mode, the LIN break character is 11 bit times long. |
| **24**<br>**RAF** | RO | Receiver Active Flag<br>When a valid start bit is detected by the receiver, this bit will assert and it will be cleared automatically when the receiver detects an idle line.<br>0b - FCUART receiver is IDLE.<br>1b - FCUART receiver is active. |
| **23**<br>**TDREF** | RO | Transmit Data Register Empty Flag<br>With the transfer FIFO enabled, when the data in FIFO is equal to or less than WATER[TXWATER], this bit will assert. With the transmit FIFO disabled, this bit will assert when the transmit DATA register is empty.<br>0b - Transmit data buffer full.<br>1b - Transmit data buffer empty. |
| **22**<br>**TCF** | RO | Transmission Complete Flag<br>TCF is set when the transfer buffer is empty and no data transfer is ongoing, and it will be cleared when there is a transmission in progress. Notice that preamble or break is also considered as transmission.<br>0b - Transfer active (sending data, preamble, or break).<br>1b - Transfer complete. |
| **21**<br>**RDRFF** | RO | Receive Data Register Full Flag<br>With receive FIFO enabled, this bit will assert when the number of data in the receive FIFO is greater than the number indicated by WATER[RXWATER]. With receive FIFO disabled, this bit is set when the DATA register is full.<br>0b - Receive data buffer empty.<br>1b - Receive data buffer full. |
| **20**<br>**IDLEF** | W1C | Idle Line Flag<br>When enough idle characters are received, this bit will be set. This bit is write one clear type (W1C). Notice that the CTRL[ITS] will have an influence on this field.<br>0b - Idle line not detected.<br>1b - Idle line was detected. |

| | | |
|---|---|---|
| **19**<br>**RORF** | W1C | Receiver Overrun Flag<br>The receiver overrun flag will be set when the receive buffer is full and new data is received. In this condition, the newly received data will be lost. After RORF is set, it needs to clear this flag to receive the new data. This register is write one clear type (W1C).<br>0b - No overrun.<br>1b - Receive overrun. |
| **18**<br>**NF** | W1C | Noise Flag<br>Noise flag will assert when the receive data bit has a glitch. This bit is write one clear type (W1C).<br>0b - No noise detected.<br>1b - Noise detected in the received data. |
| **17**<br>**FEF** | W1C | Frame Error Flag<br>Frame error flag is set when logic 0 is received where stop bit is expected. This bit is write one clear type (W1C).<br>0b - No frame error detected.<br>1b - Frame error detected. |
| **16**<br>**PEF** | W1C | Parity Error Flag<br>Parity error flash is set when the received parity bit is different with the expected parity value. This bit is write one clear type (W1C).<br>0b - No parity error.<br>1b - Parity error. |
| **15**<br>**M0F** | W1C | Match Address 0 Flag<br>Match address 0 flag is set when the received data matches MATCH[MATCH0]. This bit is write one clear type (W1C).<br>0b - Received data is not equal to MATCH0.<br>1b - Received data is equal to MATCH0. |
| **14**<br>**M1F** | W1C | Match Address 1 Flag<br>Match address 1 flag is set when the received data matches MATCH[MATCH1]. This bit is write one clear type (W1C).<br>0b - Received data is not equal to MATCH1.<br>1b - Received data is equal to MATCH1. |
| **3**<br>**RPEF** | W1C | Receive Data Parity Error Flag<br>When the parity data read out from the receive data buffer is different from the calculated parity result, the receive data parity error happens. After that, this flag will assert if PARITY_CFG[RPE] is enabled. And if PARITY_CFG[RPEIE] is enabled, an interrupt will assert. |

| | | |
|---|---|---|
| **2**<br>**TPEF** | W1C | Transmit Data Parity Error Flag<br>When the parity data read out from the transmit data buffer is different from the calculated parity result, the transmit data parity error happens this flag will assert if PARITY_CFG[TPE] enabled. And if PARITY_CFG[TPEIE] is enabled, an interrupt will assert. |

### 59.4.5 FCUART_CTRL (Control Register)

This read/write register controls various optional features of the FCUART system. This register should only be written when the transmitter and receiver are disabled.

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | R8T9 | R9T8 | TXDIR | TXINV | ORIE | NEIE | FEIE | PEIE | TIE | TCIE | RIE | IIE | TE | RE | RWC | SBK |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | M0IE | M1IE | 0 | BRKD | 7BMS | IDLECFG | | | LOOPMS | WAITEN | RXSRC | BMSEL | RSWMS | ITS | PE | PT |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCUART_CTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**R8T9** | RW | Receive Bit 8 / Transmit Bit 9<br>When reading this bit with the FCUART configured as 9-bit or 10-bit data format, it is the ninth received data bit. Read this bit before reading the DATA register.<br>When writing to this bit with the FCUART configured to 10-bit mode, it will write to the tenth data bit. Write to this bit before writing the DATA register. |
| **30**<br>**R9T8** | RW | Receive Bit 9 / Transmit Bit 8<br>When reading this bit with the FCUART configured as 10-bit data format, it is the tenth data bit received. Read this bit before reading DATA register.<br>When writing to this bit with the FCUART configured as 9-bit or 10-bit data format, it is the ninth data bit for transfer. Write to this bit before writing the DATA register. |

| | | |
|---|---|---|
| **29**<br>**TXDIR** | RW | TXD Pin Direction in Single-Wire Mode<br>Configure the TXD pin direction when FCUART is configured as single wire mode.<br>0b - TXD pin used as receive input in single-wire mode.<br>1b - TXD pin used as transmit output in single-wire mode. |
| **28**<br>**TXINV** | RW | Transmit Data Inversion<br>Configure the transmit signal polarity.<br>0b - Transmit data signal not inverted.<br>1b - Transmit data signal inverted. |
| **27**<br>**ORIE** | RW | Overrun Interrupt Enable<br>When enabled, if the overrun flag (RORF) asserts, this bit will generate an interrupt.<br>0b - Overrun interrupt disabled.<br>1b - Overrun interrupt enabled. |
| **26**<br>**NEIE** | RW | Noise Error Interrupt Enable<br>Enabling this bit will generate an interrupt when the noise flag (NF) asserts.<br>0b - Noise error interrupt disabled.<br>1b - Noise error interrupt enabled. |
| **25**<br>**FEIE** | RW | Frame Error Interrupt Enable<br>Enabling this bit will generate an interrupt if the frame error flag asserts.<br>0b - Frame error interrupt disabled.<br>1b - Frame error interrupt enabled. |
| **24**<br>**PEIE** | RW | Parity Error Interrupt Enable<br>Enabling this bit will generate an interrupt when the parity error flag (PF) asserts.<br>0b - Parity error interrupt disabled.<br>1b - Parity error interrupt enabled. |
| **23**<br>**TIE** | RW | Transmit Interrupt Enable<br>Enabling this bit will generate an interrupt when STAT[TDREF] asserts.<br>0b - Transmit interrupt disabled.<br>1b - Transmit interrupt enabled. |
| **22**<br>**TCIE** | RW | Transmission Complete Interrupt Enable<br>Enabling this bit will generate an interrupt when STAT[TCF] asserts.<br>0b - Transmit complete interrupt disabled.<br>1b - Transmit complete interrupt enabled. |
| **21**<br>**RIE** | RW | Receiver Interrupt Enable<br>Enabling his bit will generate an interrupt when STAT[RDRFF] asserts.<br>0b - Receiver interrupt disabled.<br>1b - Receiver interrupt enabled. |

| | | |
|---|---|---|
| **20**<br>**IIE** | RW | Idle Line Interrupt Enable<br>Enabling this bit will generate an interrupt when the idle line flag STAT[IDLEF] asserts.<br>0b - Idle line interrupt disabled.<br>1b - Idle line interrupt enabled. |
| **19**<br>**TE** | RW | Transmitter Enable<br>FCUART transmitter enable. When clearing and then setting TE will queue an idle preamble. This bit will remain 1 after TE is cleared until the current transmitter has been completed.<br>0b - Transmitter disabled.<br>1b - Transmitter enabled. |
| **18**<br>**RE** | RW | Receiver Enable<br>FCUART receiver enable. When RE is written to 0, it will still be read as 1 until the current receive is finished.<br>0b - Receiver disabled.<br>1b - Receiver enabled. |
| **17**<br>**RWC** | RW | Receiver Wakeup Control<br>Enabling this bit will set the FCUART receiver in a standby state waiting for a wakeup condition. When a wakeup event occurs the RWC will automatically be cleared. The CTRL[RSWMS] will determine the wakeup event. (an IDLE event or an address match)<br>Notice that if the receiver is not idle RWC must be set only with CTRL[RSWMS] cleared. It will be possible that the FCUART will discard data if the flag is set to wake up an IDLE event and the receiver is already idle. This is because the data must be received or a LIN break detected after an IDLE is detected before IDLE is allowed to be reasserted.<br>0b - FCUART receiver not configured in standby waiting for wakeup condition.<br>1b - FCUART receiver configured in standby waiting for wakeup condition. |
| **16**<br>**SBK** | RW | Send Break<br>Writing a 1 and then a 0 to SBK will queue a break character in the transmit data stream. And configuring STAT[BCGL] will impact the break character length.  Bit times of logic 0 are queued as long as SBK is set. Depending on the timing of the set and clear of SBK relative to the character currently being transmitted, a second break character may be queued before software clears SBK. A single break character can also be queued by writing to the transmit FIFO with DATA[FETSC] set and DATA[R9T9] clear.<br>0b - Normal transmitter operation.<br>1b - Queue break character(s) to be sent. |

| | | |
|---|---|---|
| **15**<br>**M0IE** | RW | Match Address 0 Interrupt Enable<br>0b - Match address 0 interrupt disabled<br>1b - Match address 0 interrupt enabled |
| **14**<br>**M1IE** | RW | Match Address 1 Interrupt Enable<br>0b - Match address 1 interrupt disabled<br>1b - Match address 1 interrupt enabled |
| **12**<br>**BRKD** | RW | Length Delimiter for LIN Break Send Out<br>0b - 1-bit length for LIN break send out<br>1b - 2-bit length for LIN break send out |
| **11**<br>**7BMS** | RW | 7-Bit Mode Select<br>Setting this bit will configure the FCUART as 7-bit data mode. Only change this bit when the transmitter and receiver are disabled.<br>0b - FCUART not configured as 7-bit data mode.<br>1b - FCUART configured as 7-bit data mode. |
| **10-8**<br>**IDLECFG** | RW | Idle Configuration<br>Configures the number of idle characters that must be received before the IDLE flag is set.<br>000b - 1 idle character<br>001b - 2 idle characters<br>010b - 4 idle characters<br>011b - 8 idle characters<br>100b - 16 idle characters<br>101b - 32 idle characters<br>110b - 64 idle characters<br>111b - 128 idle characters |
| **7**<br>**LOOPMS** | RW | Loop Mode Select<br>When LOOPMS is set, the RXD pin will not be used for the FCUART receive and the transmitter output is internally connected to the receiver input.<br>0b - Loop back mode disabled.<br>1b - Loop back mode enabled. |
| **6**<br>**WAITEN** | RW | WAIT Enable<br>0b - FCUART is enabled in wait mode.<br>1b - FCUART is disabled in wait mode. |
| **5**<br>**RXSRC** | RW | Receiver Source Select<br>If LOOPMS field is not set, setting this bit will have no effect. When LOOPMS is set, the RXSRC field determines the source for the receiver shift register input.<br>0b - With LOOPMS set, FCUART in loopback mode.<br>1b - Single-wire FCUART mode and the TXD pin is connected to the transmitter output and receiver input. |

| | | |
|---|---|---|
| **4**<br>**BMSEL** | RW | 9-Bit or 8-Bit Mode Select<br>0b - FCUART is 8-bit data characters mode.<br>1b - FCUART is 9-bit data characters mode. |
| **3**<br>**RSWMS** | RW | Receiver Wakeup Method Select<br>Select the wakeup condition of FCUART when RWC field set to 1:<br>0b - With RWC field set to 1 configure the FCUART in idle-line wakeup state.<br>1b - With RWC field set to 1 configure the FCUART in address-mark wakeup state. |
| **2**<br>**ITS** | RW | Idle Line Type Select<br>This file configures the position that receiver starts to count logic 1 as idle character bits. Either after a valid start bit or after the stop bit.<br>0b - Idle character bit count starts after start bit.<br>1b - Idle character bit count starts after stop bit. |
| **1**<br>**PE** | RW | Parity Enable<br>Enables hardware parity generation and checking.<br>0b - Parity disabled.<br>1b - Parity enabled. |
| **0**<br>**PT** | RW | Parity Type<br>When parity is enabled, this bit selects the parity generation and checking method type.<br>0b - Even parity.<br>1b - Odd parity. |

### 59.4.6 FCUART_DATA (Data Register)

This register is two separate registers. Reads return the contents of the read-only receive data buffer and writes will write to the write only transmit data buffer.

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NOISY | PARITYE | FETSC | RXEMPT | IDLE | 0 | RXTXDATA | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCUART_DATA field descriptions**

| Field | Type | Description |
|---|---|---|
| **15**<br>**NOISY** | RO | Noisy<br>This read-only bit indicates that the current data was received with noise.<br>0b - The data was received without noise.<br>1b - The data was received with noise. |
| **14**<br>**PARITYE** | RO | Parity Error<br>This read-only bit indicates that the current received data was received with a parity error.<br>0b - The data was received without a parity error.<br>1b - The data was received with a parity error. |
| **13**<br>**FETSC** | RW | Frame Error / Transmit Special Character<br>Read of this bit indicates that if the received data has a frame error.<br>Writing to this bit will queue a break or idle character to be transferred instead of the DATA[T9:T0]. When this bit writes 1, the DATA[T9] is used to indicate the transfer type. 0 means a break character, and 1 means an idle character. The contents of DATA[T8:T0] should be zero.<br>0b - Read means the data was received without a frame error, and write means transmitting a normal character.<br>1b - Read means the data was received with a frame error, and write means transmitting an idle or break character. |
| **12**<br>**RXEMPT** | RO | Receive Buffer Empty<br>This read-only bit indicates whether the receive buffer contains data.<br>0b - Receive buffer contains valid data.<br>1b - Receive buffer contains no data. |
| **11**<br>**IDLE** | RO | Idle Line<br>This read-only bit indicates whether the receiver line was idle before receiving the data.<br>0b - Receiver was not idle before receiving this character.<br>1b - Receiver was idle before receiving this character. |
| **9-0**<br>**RXTXDATA** | RW | Read receive data buffer x or write transmit data buffer x.<br>Read as R9:R0, Write as T9:T0 |

### 59.4.7 FCUART_MATCH (Match Address Register)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | MATCH1 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | MATCH0 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCUART_MATCH field descriptions**

| Field | Type | Description |
|---|---|---|
| **25-16**<br>**MATCH1** | RW | Match Address 1<br>The MATCH1 and MATCH0 fields are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the DATA register. If a match fails, the following data is discarded. Software should only write a MAx field when the associated BAUD[MAEN] bit is clear. |
| **9-0**<br>**MATCH0** | RW | Match Address 0<br>The MATCH0 and MATCH1 fields are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the DATA register. If a match fails, the following data is discarded. Software should only write a MAx field when the associated BAUD[MAEN] bit is clear. |

### 59.4.8 FCUART_MODIR (Modem IrDA Register)

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | RXRTSCFG | | 0 | | TXCTSSRC | TXCTSCFG | RXRTSEN | TXRTSPOL | TXRTSEN | TXCTSEN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCUART_MODIR field descriptions**

| Field | Type | Description |
|---|---|---|
| **9-8**<br>**RXRTSCFG** | RW | Receive RTS Configuration<br>Configures the data number that will have an influence on the assertion and negation of the RX RTS_B output. When receive FIFO has enough space to store RXRTSCFG number data, the RX RTS_B output negates. If both RX RTS_B and address/data matching are enabled, then RTS_B could assert at the end of a character if there is no match. Only set this field when the receiver is disabled. |
| **5**<br>**TXCTSSRC** | RW | Transmit CTS Source<br>Select the source of the CTS input.<br>0b - CTS input is the CTS_B pin.<br>1b - CTS input is the inverted receiver match result. |
| **4**<br>**TXCTSCFG** | RW | Transmit CTS Configuration<br>Select the CTS sample strategy.<br>0b - CTS input is sampled at the start of each character.<br>1b - CTS input is sampled when the transmitter is idle. |
| **3**<br>**RXRTSEN** | RW | Receiver Request-to-Send Enable<br>Setting this bit to 1 will use the RTS output to control the input of the transmitting device to prevent receiver overrun. Only set this bit when the receiver is disabled.<br>0b - The receiver has no effect on RTS.<br>1b - RTS function enabled. If the receiver has enough space for storing the receive data, RST will be asserted. |
| **2**<br>**TXRTSPOL** | RW | Transmitter Request-to-Send Polarity<br>Select the polarity of the transmitter RTS. this bit will not affect the receiver RTS. RTS remains negated in the active low state unless TXRTSEN is set. Only set this bit when the transmitter is disabled.<br>0b - Transmitter RTS is active low.<br>1b - Transmitter RTS is active high. |
| **1**<br>**TXRTSEN** | RW | Transmitter Request-to-Send Enable<br>Controls the RTS before and after a transfer. Only set this bit when the transmitter is disabled.<br>0b - The transmitter has no effect on RTS.<br>1b - RTS will assert one bit time before the start bit is transferred when data is written to an empty transmitter data buffer. RTS deasserts one bit time after all bit transferred. |

| | | |
|---|---|---|
| **0**<br>**TXCTSEN** | RW | Transmitter Clear-to-Send Enable<br>TXCTSEN controls the operation of the transmitter.<br>0b - CTS has no effect on the transmitter.<br>1b - Clear-to-send function enabled. If CTS is asserted, the character is sent. If CTS is deasserted, the transmission is delayed until CTS is asserted. |

### 59.4.9 FCUART_FIFO (FIFO Register)

This register provides the ability for the user to turn on and off FIFO functionality. User can also read the size of the FIFO that has been implemented. This register should only be written when transmiter and receiver are disabled and the data buffer/FIFO is empty.

Offset : 28h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TXEMPTY | RXEMPTY | 0 | | | | TXOF | RXUF |
| W | | | | | | | | | | | | | | | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | RXIDEN | | | TXOFIE | RXUFIE | TXFEN | TXFIFODEP | | | RXFEN | RXFIFODEP | | |
| W | TXFLUSH | RXFLUSH | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

**FCUART_FIFO field descriptions**

| Field | Type | Description |
|---|---|---|
| **23**<br>**TXEMPTY** | RO | Transmit Buffer/FIFO Empty<br>This read-only bit indicates whether there is no data in the Transmit FIFO/buffer.<br>0b - Transmit buffer is not empty.<br>1b - Transmit buffer is empty. |
| **22**<br>**RXEMPTY** | RO | Receive Buffer/FIFO Empty<br>This read-only bit indicates whether there is no data in the receive FIFO/Buffer.<br>0b - Receive buffer is not empty.<br>1b - Receive buffer is empty. |

| | | |
|---|---|---|
| **17**<br>**TXOF** | W1C | Transmitter Buffer Overflow Flag<br>Indicates that data was written to the transfer data buffer when it is full. This register is write one clear (W1C).<br>0b - No transmit buffer overflow has occurred.<br>1b - Transmit buffer overflow has occurred. |
| **16**<br>**RXUF** | W1C | Receiver Buffer Underflow Flag<br>Indicates that data has been read from the receive buffer when it is empty. This register is write one clear (W1C).<br>0b - No receive buffer underflow has occurred.<br>1b - Receive buffer underflow has occurred. |
| **15**<br>**TXFLUSH** | WO | Transmit FIFO/Buffer Flush<br>This write-only bit is used to flush the transmit FIFO/buffer.<br>0b - No flush operation occurs.<br>1b - All data in the transmit FIFO/Buffer is cleared out. |
| **14**<br>**RXFLUSH** | WO | Receive FIFO/Buffer Flush<br>This write-only bit is used to flush the receive FIFO/buffer.<br>0b - No flush operation occurs.<br>1b - All data in the receive FIFO/buffer is cleared out. |
| **12-10**<br>**RXIDEN** | RW | Receiver Idle Empty Enable<br>Configure the idle character number for RDRFF asserts. If configured with non zero value when FIFO is empty and the receiver is idle for a number of idle characters, the RDRFF will assert.<br>000b - Disable RDRFF assertion due to partially filled FIFO when the receiver is idle.<br>001b - Enable RDRFF assertion due to partially filled FIFO when the receiver is idle for 1 character.<br>010b - Enable RDRFF assertion due to partially filled FIFO when the receiver is idle for 2 characters.<br>011b - Enable RDRFF assertion due to partially filled FIFO when the receiver is idle for 4 characters.<br>100b - Enable RDRFF assertion due to partially filled FIFO when the receiver is idle for 8 characters.<br>101b - Enable RDRFF assertion due to partially filled FIFO when the receiver is idle for 16 characters.<br>110b - Enable RDRFF assertion due to partially filled FIFO when the receiver is idle for 32 characters.<br>111b - Enable RDRFF assertion due to partially filled FIFO when the receiver is idle for 64 characters. |

| | | |
|---|---|---|
| **9**<br>**TXOFIE** | RW | Transmit FIFO Overflow Interrupt Enable<br>When this field is set, if transfer FIFO overflow TXOF will assert and an interrupt will be generated<br>0b - Transmit FIFO overflow interrupt enabled<br>1b - Transmit FIFO overflow interrupt disabled |
| **8**<br>**RXUFIE** | RW | Receive FIFO Underflow Interrupt Enable<br>When this field is set, if receiver FIFO underflows, the RXUF flag asserts and an interrupt will be generated.<br>0b - Receive FIFO Underflow Interrupt disabled.<br>1b - Receive FIFO Underflow Interrupt Enabled. |
| **7**<br>**TXFEN** | RW | Transmit FIFO Enable<br>Enable the built-in FIFO for transmit. When enabled, TXFIFODEP field can indicate the size of the FIFO. If this field is cleared, the transmit buffer operates as a FIFO of depth one and TXFIFODEP is meaningless. Only set this bit when transmitter and receiver are disabled.<br>0b - Transmit FIFO is not enabled.<br>1b - Transmit FIFO is enabled. |
| **6-4**<br>**TXFIFODEP** | RO | Transmit FIFO Buffer Depth<br>This read-only field indicates the transfer FIFO size if the transfer FIFO is enabled.<br>000b - Transmit FIFO/Buffer depth = 1 dataword.<br>011b - Transmit FIFO/Buffer depth = 16 datawords. |
| **3**<br>**RXFEN** | RW | Receive FIFO Enable<br>Enable the built-in FIFO for receive. When enabled, RXFIFODEP field can indicate the size of the FIFO. If this field is cleared, the receive buffer operates as a FIFO of depth one and RXFIFODEP is meaningless. Only set this bit when the transmitter and receiver are disabled.<br>0b - Receive FIFO is not enabled.<br>1b - Receive FIFO is enabled. |
| **2-0**<br>**RXFIFODEP** | RO | Receive FIFO Buffer Depth<br>This read-only field indicates the receive FIFO size if the receive FIFO is enabled.<br>000b - Receive FIFO/Buffer depth = 1 dataword.<br>011b - Receive FIFO/Buffer depth = 16 datawords. |

### 59.4.10 FCUART_WATERMARK (Watermark Register)

This register can configure the water mark of the transfer or receiver FIFO. Only write to this register when transmitter and receiver are disabled.

Offset : 2Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | RXCOUNT | | | | 0 | | | | RXWATER | | | |
| W | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | | | TXCOUNT | | | | 0 | | | | TXWATER | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FCUART_WATERMARK field descriptions**

| Field | Type | Description |
|---|---|---|
| **29-24**<br>**RXCOUNT** | RO | Receive Counter<br>This read-only register indicates the data number has been received in the receive FIFO/buffer. |
| **19-16**<br>**RXWATER** | RW | Receive Watermark<br>When the number of data in the receive FIFO/buffer is greater than the value in this register field, an interrupt or a DMA request will be generated. |
| **12-8**<br>**TXCOUNT** | RO | Transmit Counter<br>This read-only register indicates the data number has been written in the transfer FIFO/buffer. |
| **3-0**<br>**TXWATER** | RW | Transmit Watermark<br>When the number of data in the transmit FIFO/buffer is equal to or less than the value in this register field, an interrupt or a DMA request will be generated. |

# Chapter 60 FLEX Controller Area Network (FLEXCAN)

## 60.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • Added a note to Section *CAN Embedded RAM Initialization;*<br>• Updated the description of the "DIRECT_TRIG" bit field in the register section *FLEXCAN_CTRL2* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 60.2 Chip-specific Information

The FC7240 series supports ten FLEXCAN instances: FLEXCAN0-FLEXCAN3. The functions supported are as follows:

| Instances | Number of MBs | CAN FD | PNET | External Time Tick | Enhanced FIFO |
|---|---|---|---|---|---|
| FLEXCAN0 | 32 | Yes | Yes | No | Yes |
| FLEXCAN1 | 32 | Yes | Yes | No | Yes |
| FLEXCAN2 | 32 | Yes | Yes | No | Yes |
| FLEXCAN3 | 32 | Yes | Yes | No | Yes |

***Note:*** Enhanced Rx FIFO which is allocated at 0x2000 - 0x23bf, can store up to 12 CAN FD messages. The region 0x2000 - 0x204F contains the output of the FIFO which should be read by the CPU or DMA. And The FC7240 series support a total of 32 filter elements allocated in 0x3000-0x307C for ID filter when Rx with Enhanced FIFO, for detailed information, please refer Enhanced Rx FIFO, Enhanced Rx (ERX) FIFO Structure and Enhanced Rx FIFO ID Match Example.

### 60.2.1 CAN Embedded RAM Initialization

To avoid ECC error, the following address must be initialized before CAN special feature is used.

| Feature Enable | RAM Region | Address Needs to Be Initialized |
|---|---|---|
| - | Message Buffers | 0x80-0x27c |
| - | RXIMRs | 0x880-0x8fc |
| - | RXFIR | 0xa80-0xa94 |
| - | MASK* | 0xaa0-0xaac* |
| - | TX_SMB | 0xab0-0xabc |
| - | RX_SMB0 | 0xac0-0xacc |
| - | RX_SMB1 | 0xad0-0xadc |
| CAN FD | TX_SMB | 0xf28-0xf6c |
| CAN FD | RX_SMB0 | 0xf70-0xfb4 |
| CAN FD | RX_SMB1 | 0xfb8-0xffc |
| Enhanced FIFO | ERX_FIFO | 0x2000-0x23bc |
| Enhanced FIFO | ERFFEL | 0x3000-0x307c |

***Note:*** MASK here indicates RX14MASK, RX15MASK, RXMGMAXK, and RSFGMASK, and the address 0xaa0 - 0xaac is only valid when RAM is initialized.

## 60.3 Features

The FLEX Controller Area Network (FLEXCAN) module includes the following distinctive features:

- Full implementation of the CAN FD protocol and *CAN Specification* 2.0, Part B
  - Standard data frames
  - Extended data frames
  - 0 to 64 bytes data length
  - Programmable bit rate
  - Content-related addressing
- Compliant with the ISO 11898-1 standard
- Flexible mailboxes configurable to store 0 to 8, 16, 32, or 64 bytes data length
- Each mailbox configurable as receive or transmit, all supporting standard and extended messages
- Individual Rx Mask registers per mailbox

- Full-featured Rx FIFO with storage capacity for up to six frames and automatic internal pointer handling with DMA support
- Transmission abort capability
- Flexible message buffers (MBs), totaling 32 message buffers of 8 bytes data length each, configurable as Rx or Tx
- Programmable clock source to the CAN Protocol Interface, either peripheral clock or oscillator clock
- RAM not used by reception or transmission structures can be used as general-purpose RAM space
- Listen-only mode capability
- Programmable Loop-Back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Maskable interrupts
- Independence from the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes or matching with received frames (Pretended Networking)
- Transceiver delay compensation feature when transmitting CAN FD messages at faster data rates
- Remote request frames may be managed automatically or by software
- CAN bit time settings and configuration bits can only be written in Freeze mode
- Tx mailbox status (Lowest priority buffer or empty buffer)
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- SYNCH bit available in Error in Status 1 register to indicate that the module is synchronous with FLEXCAN bus
- CRC status for the transmitted message
- Rx FIFO Global Mask register
- Selectable priority between mailboxes and Rx FIFO during the matching process
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard, or 512 partial (8 bit) IDs, with up to 32 individual masking capability
- Support detection and correction of errors in memory read accesses. Each byte of FLEXCAN memory is associated with 5 parity bits. The error correction mechanism ensures that in this 13-bit word, errors in one bit can be corrected (correctable errors) and errors in 2 bits can be detected but not corrected (non-correctable errors).
- Support Pretended Networking functionality in low power: stop mode.
- Support the direct CAN connection function, allowing to connect multiple CANs without PHY

## 60.4 Overview

The FLEXCAN module is a communication controller implementing the CAN protocol according to the ISO 11898-1 standard and CAN 2.0 B protocol specifications, which includes:

- One associated memory for storing message buffers (MBs)
- Receive Global Mask registers
- Receive Individual Mask registers
- Receive FIFO filters
- Receive FIFO ID filters

### 60.4.1 Modes of Operation

The FLEXCAN module has the following functional modes:

**Normal mode (User or Supervisor):**

In Normal mode, the module operates receiving and/or transmitting message frames, errors are managed normally, and all CAN Protocol functions are enabled. User and Supervisor modes differ in the access to some restricted control registers.

**Freeze mode:**

Freeze mode is enabled when MCR[FRZ] is asserted. If enabled, Freeze mode is entered when MCR[HALT] is set or when Debug mode is requested at the chip level and MCR [FRZ_ACK] is asserted by the FLEXCAN. In this mode, no transmission or reception of frames is done and synchronicity to the FLEXCAN bus is lost.

**Loop-Back mode:**

The module enters this mode when CTRL1[LPB] is asserted. In this mode, FLEXCAN performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx FLEXCAN input pin is ignored and the Tx FLEXCAN output goes to the recessive state (logic '1').

FLEXCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FLEXCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

**Listen-Only mode:**

The module enters this mode when CTRL1[LOM] is asserted. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a FLEXCAN Error Passive mode. Only messages acknowledged by another FLEXCAN station will be received. If FLEXCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

**CAN FD Active mode:**

In this mode, FLEXCAN is capable of transmitting and receiving all messages formatted according to the CAN FD Protocol and CAN 2.0 Protocol 2.0 in an interleaved fashion. The CPU can set the FLEXCAN into CAN FD Active mode by configuring MCR[FDEN] in Freeze mode.

**Module Disable mode:**

This low-power mode is entered when MCR[MDIS] is asserted by the CPU and MCR[LPM_ACK] is asserted by FLEXCAN. When disabled, the module issues a request to disable the clocks to the CAN Protocol Engine and Controller Host Interface submodules. Exit from this mode is done by negating MCR[MDIS]. See Module Disable mode for more information.

**Stop mode:**

This low power mode is entered when Stop mode is requested at the chip level and MCR[LPM_ACK] is asserted by the FLEXCAN. When in Stop mode, the module puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally. Exit from this mode happens when the Stop mode request is removed. See Stop mode for more information.

**Pretended Network mode:**

This mode can be selected to operate together with Stop mode. Before entering one of these low power modes, you must assert MCR[PNET_EN]. When in low power mode, CHI sub-block clocks are shut down and FLEXCAN_PE sub-block is kept clocked, so that the Rx receive process is still active to filter incoming messages (see Receive process under Pretended Networking mode) as defined by the configuration registers (see Pretended Networking Control 1 Register (CTRL1_PN)). Upon detecting a wake-up event, a wake-up interrupt is issued to the system.

### 60.4.2 Signal Descriptions

The FLEXCAN module has two I/O signals connected to the external chip pins. These signals are summarized in the following table and described in more detail in the next subsections.

**Table 90. FLEXCAN I/O signals**

| Signal | Description | I/O |
|---|---|---|
| FLEXCAN Rx | FLEXCAN receive pin | Input |
| FLEXCAN Tx | FLEXCAN transmit pin | Output |

**FLEXCAN Rx**

This pin is the receive pin from the FLEXCAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

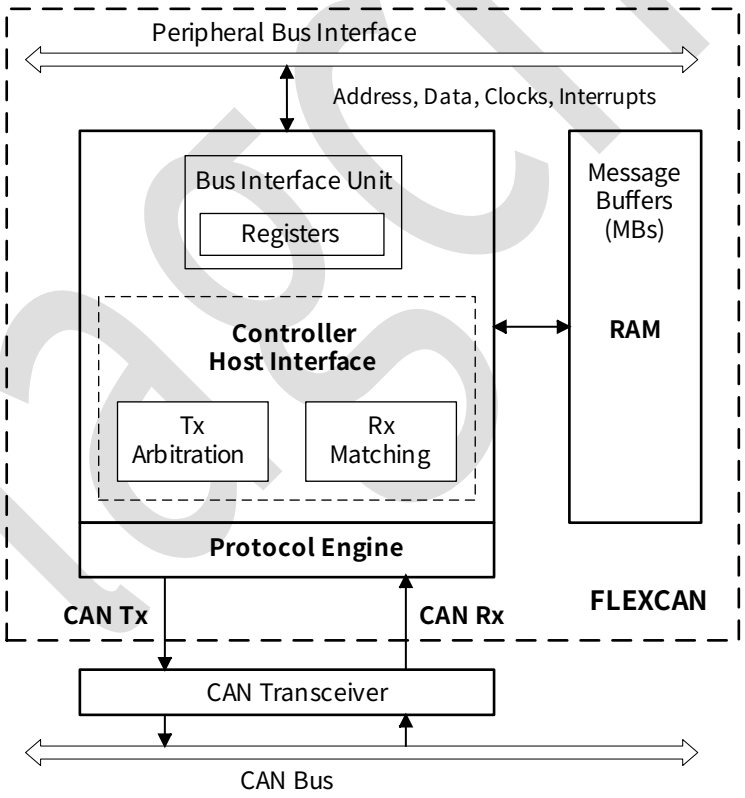
**FLEXCAN Tx**

This pin is the transmit pin to the FLEXCAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

## 60.5 Block Diagram

The following is the block diagram of the FLEXCAN module.

**Figure 139. FLEXCAN block diagram**




## 60.6 Functional Description

The FLEXCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed of a set of message buffers (MB) that store configuration and control data, time stamp, message ID, and data. The memory corresponding to the first 32 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 128 extended IDs or 256 standard IDs or 512 8-bit ID slices), with individual mask register for up to 32 ID filter table elements.

For Classical CAN frames, simultaneous reception through FIFO and mailbox is supported. For CAN FD frames, reception is supported through mailboxes only. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed in the ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A message buffer is said to be active at a given time if it can participate in both the matching and arbitration processes. An Rx MB with a 0b0000 code is inactive. Similarly, a Tx MB with a 0b1000 or 0b1001 code is also inactive.

The FLEXCAN module is also able to receive and transmit messages in CAN FD format. The message buffers are sized to adequately store the quantity of data bytes selected by the FDCTRL[MBDSRn] fields. The quantity of FD MBs available for a given quantity of data bytes is described in the FDCTRL register.

### 60.6.1 Initialization Sequence

The FLEXCAN module may be reset in two ways:

1) Chip level hard reset, which resets all memory-mapped registers asynchronously.
2) MCR[SOFTRST], which resets some of the memory-mapped registers synchronously.

A soft reset is synchronous and must follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. MCR[SOFTRST] remains asserted while soft reset is pending, so the software can poll this bit to know when the reset has completed. Also, the soft reset cannot be applied while clocks are shut down in a low-power mode. The low-power mode should be exited, and the clocks resumed before applying the soft reset.

The clock source should be selected when the module is in Disable mode. After the clock source is selected and the module is enabled (MCR[MDIS] bit negated), FLEXCAN automatically enters Freeze mode. In Freeze mode, FLEXCAN is un-synchronized to the FLEXCAN bus, MCR[HALT] and MCR[FRZ] are set, and the internal state machines are disabled, then MCR[FRZACK] and MCR[NOTRDY] are set. The Tx pin is in a recessive state and FLEXCAN does not initiate any transmission or reception of CAN frames. Note that the message buffers and the Rx Individual Mask registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization, it is required that FLEXCAN should be put into Freeze mode (see Freeze mode). The following is a generic initialization sequence applicable to the FLEXCAN module:

1) Initialize the Module Configuration Register (MCR).
   a) Enable the individual filtering per MB and reception queue features by setting IRMQ.
   b) Enable the warning interrupts by setting WRNEN.
   c) If required, disable frame self-reception by setting SRXDIS.
   d) Enable the Rx FIFO by setting RFEN.
   e) If Rx FIFO is enabled and DMA is required, set DMA.
   f) If Pretended Networking mode is required, set PNET_EN.
   g) Enable the abort mechanism by setting AEN.
   h) Enable the local priority feature by setting LPRIOEN.
2) Initialize the Control 1 register (CTRL1) and optionally the CAN Bit Timing register (CBT). Also initialize the CAN FD Bit Timing register (FDCBT).
   a) Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, and RJW.
   b) Optionally determine the bit timing parameters: EPROPSEG, EPSEG1, EPSEG2, and ERJW.

c) Determine the CAN FD bit timing parameters: FPROPSEG, FPSEG1, FPSEG2, and FRJW.
d) Determine the bit rate by programming the PRESDIV field and optionally the EPRESDIV field.
e) Determine the CAN FD bit rate by programming the FPRESDIV field.
f) Determine the internal arbitration mode (LBUF).

3) Initialize the message buffers.
   a) The control and status word of all message buffers must be initialized.
   b) If Rx FIFO was enabled, the ID filter table must be initialized.
   c) Other entries in each message buffer should be initialized as required.
4) Initialize the Rx Individual Mask registers (RXIMRn).
5) Set required interrupt mask bits in:
   a) IMASK registers (for all MB interrupts)
   b) CTRL1 / CTRL2 registers (for Bus Off and Error interrupts)
6) If Pretended Networking mode is enabled, configure the necessary registers for selective wakeup.
7) Negate MCR[HALT].

After the last step above is completed, FLEXCAN attempts to synchronize to the FLEXCAN bus.

### 60.6.2 Transmit Process

To transmit a CAN frame, the CPU must prepare a message buffer for transmission by executing the following procedure:

1) Check whether the respective interrupt bit is set and clear it.
2) If the MB is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control and Status word to request an abort of the transmission.
3) Wait for the corresponding IFLAG bit to be asserted by polling the IFLAG register, or by the interrupt request if enabled by the respective IMASK bit.
4) Read back the CODE field to check if the transmission was aborted or transmitted.
5) Clear the corresponding interrupt flag.
6) Write the ID register (containing the local priority if enabled via MCR[LPRIO_EN]).
7) Write payload data bytes.
8) Configure the Control and Status word with the desired configuration.
   a) Set ID type via MB_CS[IDE].
   b) Set Remote Transmission Request (if needed) via MB_CS[RTR].
   c) If MCR[FDEN] is enabled, also configure the MB_CS[EDL] and MB_CS[BRS] fields. For details about the relationship between the written value and transmitted value of the MB_CS[ESI] field.
   d) Set Data Length Code in bytes via MB_CS[DLC].
   e) Activate the message buffer to transmit the CAN frame by setting MB_CS[CODE] to 0xC.

***Note:*** It is strongly recommended that all the fields in MB_CS word be configured in only one 32-bit write operation to maximize software performance. If the fields are configured in separate writes, the MB_CS[CODE] must be the last write in the C/S word.

When the MB is activated, it participates in the arbitration process and is eventually transmitted according to its priority. When the DLC value stored in the MB selected for transmission is larger than the respective MB payload size, FLEXCAN adds the necessary number of bytes with a constant 0xCC pattern to complete the expected DLC.

At the end of the successful transmission:

- The value of the free-running timer is written into the Time Stamp field.

- The CODE field in the Control and Status word is updated.
- Both CRC and FDCRC registers are updated.
- A status flag is set in the Interrupt Flag register.
- An interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The new CODE field after transmission depends on the code used to activate the MB.

When the Abort feature is enabled (MCR[AEN] is asserted), after the Interrupt flag is asserted for a mailbox configured as transmit buffer, the mailbox is blocked. Therefore, the CPU is not able to update it until the Interrupt Flag is negated by the CPU. This means that the CPU must clear the corresponding IFLAG bit before starting to prepare this MB for a new transmission or reception.

***Note:*** If backwards compatibility is desired (MCR[AEN] is negated), write the INACTIVE code (0b1000) to the CODE field to inactivate the MB. However, in this case, the pending frame may be transmitted without notification (see Mailbox inactivation).

### 60.6.3 Arbitration Process

The arbitration process scans the mailboxes, searching for the Tx mailbox that holds the message to be sent in the next opportunity. This mailbox is called the arbitration winner.

The scan starts from the lowest number mailbox and runs toward the higher ones. The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the value of CTRL2[TASD].
- During the Error Delimiter field of a CAN frame.
- During the Overload Delimiter field of a CAN frame.
- When the winner is inactivated and the FLEXCAN bus has still not reached the first bit of the Intermission field.
- When there is a CPU write to the C/S word of a winner MB and the FLEXCAN bus has still not reached the first bit of the Intermission field.
- When CHI is in Idle state and the CPU writes to the C/S word of any MB.
- When FLEXCAN exits Bus Off state.
- Upon leaving Freeze mode or Low Power mode.

If the arbitration process does not manage to evaluate all mailboxes before the FLEXCAN bus has reached the first bit of the Intermission field, the temporary arbitration winner is invalidated and the FLEXCAN will not compete for the FLEXCAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx mailboxes at the end of the scan according to the settings of both CTRL1[LBUF] and MCR[LPRIOEN].

If CTRL1[LBUF] is asserted, the first (lowest number) active Tx mailbox found is the arbitration winner. MCR[LPRIOEN] has no effect when CTRL1[LBUF] is asserted.

After the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user-accessible. This operation is called move-out and after it is done, write access to the C/S word of the corresponding MB is blocked (if MCR[AEN] is asserted). Write access is restored in the following events:

- After the MB is transmitted and the corresponding IFLAG bit is cleared by the CPU.
- FLEXCAN enters Freeze mode or Bus Off.
- FLEXCAN loses the bus arbitration or there is an error during the transmission.

At the first opportunity window on the FLEXCAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules.

- Arbitration process can be triggered in the following situations:
- During Rx and Tx frames from FLEXCAN CRC field to end of frame. CTRL2[TASD] value may be changed to optimize the arbitration start point.
- During FLEXCAN Bus Off state from TX_ERR_CNT=124 to 128. CTRL2[TASD] value may be changed to optimize the arbitration start point.
- During C/S write by the CPU in Bus Idle. First C/S write starts the arbitration process, and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, Tx arbitration process is pending. If there is no arbitration winner after the arbitration process has finished, then the TX arbitration machine begins a new arbitration process. If there is a pending arbitration and Bus Idle state starts, then an arbitration process is triggered. In this case, the first and second C/S write in Bus Idle will not restart the arbitration process. It is possible that there is not enough time
- To finish arbitration in Wait For Bus Idle state and the next state is Idle. In this case the scan is not interrupted, and it is completed during Bus Idle state. During this arbitration, C/S write does not cause arbitration restart.
- Arbitration winner deactivation during a valid arbitration window.
- Upon exiting Freeze mode (first bit of the Wait For Bus Idle state). If there is a re-synchronization during Wait For Bus Idle, the arbitration process is restarted.

The arbitration process stops in the following situations:

- All mailboxes were scanned.
- A Tx active mailbox is found if the lowest buffer feature is enabled.
- Arbitration winner inactivation or abort during any arbitration process.
- There was not enough time to finish Tx arbitration process (for instance, when a deactivation was performed near the end of frame). In this case, the arbitration process is pending.
- Error or overload flag in the bus.
- Low Power or Freeze mode request in Idle state.

Arbitration is considered pending as described below:

- It was not possible to finish the arbitration process in time.
- C/S write during arbitration if a write is performed in an MB whose number is lower than the Tx arbitration pointer.
- Any C/S write if there is no Tx arbitration process in progress.
- Rx Match has just updated an Rx code to Tx code.
- Entering Bus Off state.

C/S write during arbitration has the following effect:

- If C/S write is performed in the arbitration winner, a new process is restarted immediately.
- If C/S write is performed in an MB whose number is higher than the Tx arbitration pointer, the ongoing arbitration process will scan this MB as normal.

#### *60.6.3.1 Highest-priority Mailbox First*

If CTRL1[LBUF] is negated, then the arbitration process searches the active Tx mailbox with the highest priority, which means that this mailbox's frame would have a higher probability to win the arbitration on FLEXCAN bus when multiple external nodes compete for the bus at the same time.

The sequence of bits considered for this arbitration is called the arbitration value of the mailbox. The highest-priority Tx mailbox is the one that has the lowest arbitration value among all Tx mailboxes.

If two or more mailboxes have equivalent arbitration values, the mailbox with the lowest number is the arbitration winner.

The composition of the arbitration value depends on the MCR[LPRIOEN] setting.

#### *60.6.3.2 Local Priority Disabled*

If MCR[LPRIOEN] is negated the arbitration value is built in the exact sequence of bits as they would be transmitted in a CAN frame (see the following table) in such a way that the local priority is disabled.

**Table 91. Composition of the arbitration value when local priority is disabled**

| Format | Mailbox Arbitration Value (32 bits) | | | | |
|---|---|---|---|---|---|
| Standard (IDE = 0) | Standard ID (11 bits) | RTR (1 bit) | IDE (1 bit) | - (18 bits) | - (1 bit) |
| Extended (IDE = 1) | Extended ID[28:18] (11 bits) | SRR (1 bit) | IDE (1 bit) | Extended ID[17:0] (18 bits) | RTR (1 bit) |

#### *60.6.3.3 Local Priority Enabled*

To enable local priority, MCR[LPRIOEN] must be asserted. In this case, the mailbox PRIO field is included at the very left of the arbitration value (see the following table).

**Table 92. Composition of the arbitration value when local priority is disabled**

| Format | Mailbox Arbitration Value (35 bits) | | | | | |
|---|---|---|---|---|---|---|
| Standard(IDE= 0) | PRIO (3 bits) | Standard ID (11 bits) | RTR (1 bit) | IDE (1 bit) | - (18 bits) | - (1 bit) |
| Extended(IDE= 1) | PRIO (3 bits) | Extended ID[28:18] (11 bits) | SRR (1 bit) | IDE (1 bit) | Extended ID[17:0] (18 bits) | RTR (1 bit) |

Because the PRIO field is the most significant part of the arbitration value, mailboxes with low PRIO values have higher priority than mailboxes with high PRIO values regardless of the rest of their arbitration values.

Note that the PRIO field is not part of the frame on the FLEXCAN bus. Its purpose is only to affect the internal arbitration process.

### 60.6.4 Receive Process

To be able to receive CAN frames into a mailbox, the CPU must prepare it for reception by executing the following steps:

1) If the mailbox is active (either Tx or Rx), inactivate the mailbox (see Mailbox inactivation), preferably with a safe inactivation (see Transmission abort mechanism).
2) Write the ID word.
3) Write the EMPTY code (0b0100) to the CODE field of the Control and Status word to activate the mailbox. No setup is required for EDL, BRS, and ESI bits—they are overwritten by the respective bit fields in the received message.

After the MB is activated, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the mailbox is updated by the move-in process (see Move-in) as follows:

1) The received data field (8 bytes at most for Classical CAN message format and up to 64 bytes for CAN FD message format) is stored.
2) The received Identifier field is stored.
3) The value of the Free-Running Timer at the time of the second bit of frame's Identifier field is written into the mailbox's Time Stamp field.
4) The received SRR, IDE, RTR, EDL, BRS, ESI, and DLC fields are stored.
5) The CODE field in the Control and Status word is updated.
6) A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The recommended way for the CPU to service (read) the frame received in a mailbox is by the following procedure:

1) Read the Control and Status word of that mailbox.
2) Check if the BUSY bit is de-asserted, indicating that the mailbox is locked. Repeat step 1) while it is asserted. See Mailbox lock mechanism.
3) Read the contents of the mailbox. After the mailbox is locked, its contents won't be modified by FLEXCAN move-in processes. See Move-in.
4) Acknowledge the proper flag at IFLAG registers.
5) Read the free-running timer to unlock the mailbox.

The CPU should poll for frame reception by the status flag bit for the specific mailbox in one of the IFLAG registers and not by the CODE field of that mailbox. Polling the CODE field does not work because after a frame is received and the CPU services the mailbox (by reading the C/S word followed by unlocking the mailbox), the CODE field will not return to EMPTY. It will remain FULL. If the CPU tries to work around this behavior by writing to the C/S word to force an EMPTY code after reading the mailbox without a prior safe inactivation, a newly received frame matching the filter of that mailbox may be lost.

***Note:*** In summary: never do polling by reading directly the C/S word of the mailboxes. Instead, read the IFLAG registers.

Note that the received frame's Identifier field is always stored in the matching mailbox, thus the contents of the ID field in a mailbox may change if the match was due to masking. When MCR[SRXDIS] is asserted, FLEXCAN will not store frames transmitted by itself in any MB, even if it contains a matching Rx mailbox, and no interrupt flag or interrupt signal will be generated. Otherwise, when MCR[SRXDIS] is deasserted, FLEXCAN can receive frames transmitted by itself if a matching Rx mailbox exists.

To be able to receive CAN frames through the Rx FIFO, the CPU must enable and configure the Rx FIFO during Freeze mode (see Rx FIFO). Upon receiving the Frames Available in Rx FIFO interrupt (see the description of IFLAG1[BIT5] "Frames available in Rx FIFO"), the CPU should service the received frame using the following procedure:

1) Read the Control and Status word (optional: needed only if a mask was used for IDE and RTR bits).
2) Read the ID field (optional: needed only if a mask was used).
3) Read the data field.
4) Read the RXFIR register (optional).
5) Clear the Frames Available in Rx FIFO interrupt by writing one to IFLAG1[BIT5] (mandatory: releases the MB and allows the CPU to read the next Rx FIFO entry).

When MCR[DMA] is asserted, upon receiving a frame in FIFO, IFLAG1[BIT5] generates a DMA request and does not generate a CPU interrupt (see Rx FIFO under DMA operation). The IMASK1 bits in Rx FIFO region are not used.

The DMA controller must service the received frame using the following procedure:

1) Read the Control and Status word (read 0x80 address, optional).
2) Read the ID field (read 0x84 address, optional).
3) Read all data bytes (start read at 0x88 address, optional).
4) Read the last data bytes (read 0x8C address is mandatory).

### 60.6.5 Mask of ID Filter

When the MCR[IRMQ] bit is negated, RXMGMASK is used to mask the filter fields of all Rx MBs (the bits in the MG field will mask the mailbox filter in corresponding bits), excluding MBs 14–15 which have individual mask registers RX14MASK and RX15MASK. The Mask mechanism is shown as follows.

**Table 93. Mask mechanism**

| SMB (0) [RTR] (1) | CTRL2[RRS] | CTRL2[EACEN] | Mailbox Filter Fields | | | |
|---|---|---|---|---|---|---|
| | | | MB[RTR] | MB[IDE] | MB[ID] (4) | Reserved |
| 0 | - | 0 | Note (2) | Note (3) | MG [28:0] | MG [31:29] |
| 0 | - | 1 | MG [31] | MG [30] | MG [28:0] | MG [29] |
| 1 | 0 | - | - | - | - | MG [31:0] |
| 1 | 1 | 0 | - | - | MG [28:0] | MG [31:29] |
| 1 | 1 | 1 | MG [31] | MG[30] | MG [28:0] | MG [29] |

***Note:***

- 0 indicates the SMB definition shown in Move Process Section.
- 1 indicates RTR bit of the incoming frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
- 2 indicates if CTRL2.EACEN is negated, the RTR bit of the mailbox is never compared with the RTR bit of the incoming frame.
- 3 indicates if CTRL2.EACEN is negated, the IDE bit of the mailbox is always compared with the IDE bit of the incoming frame.
- 4 indicates the Received ID stored in SMB[ID] ([28:18] is defined as standard ID, and [28:0] defined as extends ID. The move-in process is the copy of a message received by an Rx SMB to an Rx mailbox or FIFO that has matched it).

When the MCR[IRMQ] bit is asserted, the Rx Individual Mask Registers (RXIMRn) are used to store the acceptance masks for ID filtering in Rx MBs and the Legacy Rx FIFO. For example, RXIMR0 stores the individual mask associated with either MB0 or ID Filter Table Element 0, and RXIMR1 stores the individual mask associated with either MB1 or ID Filter Table Element 1, and so on.

When the Legacy Rx FIFO is disabled (MCR[RFEN] is negated), an individual mask is provided for each available Rx mailbox on a one-to-one correspondence. When the Legacy Rx FIFO is enabled (MCR[RFEN] is asserted), an individual mask is provided for each Legacy Rx FIFO ID Filter Table Element on a one-to-one correspondence depending on the setting of CTRL2[RFFN]. RXFGMASK is used to mask the Legacy Rx FIFOID Filter Table elements that do not have a corresponding RXIMR according to FLEXCAN_CTRL2[RFFN] field settings.

This table shows how the Rx FIFO filter structure is determined by the value of CTRL2[RFFN]. See the CTRL2[RFFN] field description for more information.

**Table 94. Number of legacy Rx FIFO filters**

| RFFN [3:0] | Number of Rx FIFO Filter Elements | Message Buffers Occupied by Rx FIFO and ID Filter Table | Remaining Available Mailboxes | Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks | Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask |
|---|---|---|---|---|---|
| 0x0 | 8 | MB 0-7 | MB 8-31 | Elements 0-7 | none |
| 0x1 | 16 | MB 0-9 | MB 10-31 | Elements 0-9 | Elements 10-15 |
| 0x2 | 24 | MB 0-11 | MB 12-31 | Elements 0-11 | Elements 12-23 |
| 0x3 | 32 | MB 0-13 | MB 14-31 | Elements 0-13 | Elements 14-31 |
| 0x4 | 40 | MB 0-15 | MB 16-31 | Elements 0-15 | Elements 16-39 |
| 0x5 | 48 | MB 0-17 | MB 18-31 | Elements 0-17 | Elements 18-47 |
| 0x6 | 56 | MB 0-19 | MB 20-31 | Elements 0-19 | Elements 20-55 |
| 0x7 | 64 | MB 0-21 | MB 22-31 | Elements 0-21 | Elements 22-63 |
| 0x8 | 72 | MB 0-23 | MB 24-31 | Elements 0-23 | Elements 24-71 |
| 0x9 | 80 | MB 0-25 | MB 26-31 | Elements 0-25 | Elements 26-79 |
| 0xA | 88 | MB 0-27 | MB 28-31 | Elements 0-27 | Elements 28-87 |
| 0xB | 96 | MB 0-29 | MB 30-31 | Elements 0-29 | Elements 30-95 |
| 0xC | 104 | MB 0-31 | none | Elements 0-31 | Elements 32-103 |

### 60.6.6 Matching Process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the FLEXCAN bus. If the FIFO is enabled, the priority of scanning can be selected between mailboxes and FIFO filters. The matching starts from the lowest number message buffer toward the higher ones. If no match is found within the first structure then the other is scanned subsequently. If the FIFO is full, the matching algorithm always looks for a matching MB outside the FIFO region.

For Rx FIFO see Rx FIFO.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- If the received frame is a remote frame, the start point is the CRC field of the frame.
- If the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame.
- If the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame.

If a matching ID is found in the FIFO table or in one of the mailboxes, the contents of the Rx SMB are transferred to the FIFO or to the matched mailbox by the move-in process. If any CAN protocol error is detected, then no match results are transferred to the FIFO or to the matched mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and the active Rx mailboxes (CODE is EMPTY, FULL, OVERRUN, or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the FLEXCAN bus. The Rx SMB has the same structure as a mailbox. The reception structures (Rx FIFO or mailboxes) associated with the matching elements that had a successful comparison are the matched structures. The matching winner is selected at the end of the scan among those matched structures and depends on conditions described ahead.

A reception structure is free-to-receive when any of the following conditions is satisfied:

- The CODE field of the mailbox is EMPTY.
- The CODE field of the mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the CPU and unlocked as described in Mailbox lock mechanism).
- The CODE field of the mailbox is either FULL or OVERRUN and an inactivation (see Mailbox inactivation) is performed.
- The Rx FIFO is not full.

The scan order for mailboxes and Rx FIFO is from the matching element with the lowest number to the higher ones.

The matching winner search for mailboxes is affected by MCR[IRMQ]. If it is negated, the matching winner is the first matched mailbox regardless if it is free-to-receive or not. If it is asserted, the matching winner is selected according to the priority below:

1) the first free-to-receive matched mailbox;
2) the last non free-to-receive matched mailbox.

It is possible to select the priority of scan between mailboxes and Rx FIFO with CTRL2[MRP].

If the selected priority is Rx FIFO first:

- If the Rx FIFO is a matched structure and is free-to-receive, then the Rx FIFO is the matching winner regardless of the scan for mailboxes.
- Otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched among mailboxes as described above.
- If the selected priority is mailboxes first:
- If a free-to-receive matched mailbox is found, it is the matching winner regardless of the scan for Rx FIFO.
- If no matched mailbox is found, then the matching winner is searched in the scan for the Rx FIFO.
- If both conditions above are not satisfied and a non–free-to-receive matched mailbox is found, then the matching winner determination is conditioned by MCR[IRMQ]:
- If MCR[IRMQ] is negated, the matching winner is the first matched mailbox.
- If MCR[IRMQ] is asserted, the matching winner is the Rx FIFO if it is a free-to-receive matched structure; otherwise, the matching winner is the last non–free- to-receive matched mailbox.

If a non-safe mailbox inactivation (see Mailbox inactivation) occurs during the matching process and the mailbox inactivated is the temporary matching winner, then the temporary matching winner is invalidated. The matching elements scan is not stopped nor restarted—it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the FIFO is disabled, IRMQ is enabled, there are two MBs with the same ID, and FLEXCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm finds the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm finds MB number 2 again, but it is not "free-to-receive", so it keeps looking, finds MB number 5 and stores the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive", so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID,

received messages are queued into the MBs. The CPU can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID acceptance masks. FLEXCAN supports individual masking per MB (see the description of Rx Individual Mask registers (RXIMRn)). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is a "don't care." Note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed when the module is in Freeze mode; otherwise, they are blocked by hardware.

FLEXCAN also supports an alternate masking scheme with only four mask registers (RXFGMASK, RXMGMASK, RX14MASK, and RX15MASK) for backwards compatibility with legacy applications. This alternate masking scheme is enabled when the IRMQ bit in the MCR register is negated.

### 60.6.7 Receive Process under Pretended Networking Mode

Pretended Networking mode adds specific wakeup functionality in low power mode (Stop mode). When Pretended Networking mode (PN) is enabled by asserting MCR[PNET_EN], FLEXCAN continues processing Rx CAN messages under low power mode, and is able to detect specific wakeup messages by filtering them against ID and payload target values using preselected matching criteria. Wakeup functionality is not available for messages in CAN FD format. When in Pretended Networking mode, CAN FD format messages are ignored.

PN registers are located in the 0x0B00–0B7C address range and can be written only in Freeze mode. These registers are used for writing PN configuration (both control and target values) prior to entering Pretended Networking mode, and for reading wakeup flags and the received message ID and data when returning to normal mode after wakeup. The CPU must wait for MCR[LPMACK] to be negated before performing any access to FLEXCAN PN registers.

PN control registers are described in Pretended Networking Control 1 Register (CTRL1_PN) and Pretended Networking Control 2 Register (CTRL2_PN). The control bit fields that configure the filtering criteria are:

- PLFS: payload filtering selection
- IDFS: ID filtering selection
- FCS: filtering combination selection

PN target values are:

- FLT_IDE: IDE target value used to filter the incoming message by its format (standard or extended).
- FLT_RTR: RTR target value used to filter the incoming message by its type (data or remote frame).
- FLT_DLC_HI and FLT_DLC_LO: target DLC range used to filter the size of the payload part of an incoming message.
- FLT_ID1: ID target value used to filter the incoming message ID (equal to, smaller than or equal, greater than or equal, or the lower limit value in an ID range).
- FLT_ID2: ID target value used as the upper limit in an ID range.
- PL1: Payload target value used to filter the incoming message payload (equal to, smaller than or equal to, greater than or equal to, or the lower limit value in a payload range).
- PL2: Payload target value used as the upper limit in a payload range.

IDE, RTR, ID, and payload filters have their respective masks. These masks determine which bits are considered in equality comparisons ("1's" in certain mask positions) and which ones are don't care ("0's" in other mask positions). ID and payload masks are used only for exact ID and/or exact payload comparisons.

The ID of Rx incoming messages can be filtered based on the following criteria:

- A match with the exact ID value by detecting the equality between the ID field of the incoming message and the content of the target FLT_ID1 register. The ID mask is used.
- A match with the maximum range of ID, that is, any message with ID value smaller than or equal to the content of the target FLT_ID1 register is accepted. The ID mask is not used.
- A match with the minimum range of ID, that is, any message with ID value greater than or equal to the content of the target FLT_ID1 register is accepted. The ID mask is not used.
- A match inside a range of IDs, that is, any message with an ID value that is greater than or equal to the content of the target FLT_ID1 register and smaller than or equal to the content of target FLT_ID2_IDMASK register is accepted. The ID mask is not used.

See CTRL1_PN[IDFS] in Pretended Networking Control 1 register (FLEXCAN_CTRL1_PN).

The above criteria for ID filtering must be coherent with FLT_IDE and FLT_RTR target values in the FLT_ID1 register. Only Rx frames that match the respective IDE and RTR bits to the contents of FLT_IDE and FLT_RTR bit fields will be compared. When a range of IDs is selected (CTRL1_PN[IDFS] = 11), both FLT_ID1 and FLT_ID2 are referred to the same FLT_IDE and FLT_RTR bits in FLT_ID1 register.

The ID mask is applied only to the exact ID comparison filtering option (CTRL1_PN[IDFS] = 00) to determine which bits are considered in comparison. For the exact match option, the mask can select any bit within the ID field. For maximum range, minimum range, and inside range comparisons, the ID mask is not considered.

The IDE and RTR masks are applied in both exact and range ID comparison filtering options to determine which bits are considered in comparison.

Similar to the ID criteria, 64-bit data or payloads (PL) of Rx incoming messages can be filtered based on the following criteria:

- A match with the exact payload value by detecting the equality between the payload field of the incoming message and the content of the PL1 register. The payload mask is used.
- A match with the maximum range of payload, that is, any message with payload value smaller than or equal to the content of PL1 register is accepted. The payload mask is not used.
- A match with the minimum range of payload, that is, any message with payload value greater than or equal to the content of PL1 register is accepted. The payload mask is not used.
- A match inside a range of payloads, that is, any message with a payload value that is greater than or equal to the content of PL1 register and smaller than or equal to the content of PL2 register is accepted. The payload mask is not used.

See CTRL1_PN[PLFS] in Pretended Networking Control 1 register (FLEXCAN_CTRL1_PN).

The above criteria for payload filtering must be coherent with FLT_DLC upper and lower limit values in the FLT_DLC register. The payload of an Rx incoming message is filtered by the selected criteria only if the DLC value of the Rx incoming message is inside a DLC range:

- Greater than or equal to the FLT_DLC_LO (lower limit)
- Lower than or equal to the FLT_DLC_HI (upper limit)

Conversely, a DLC value out of the specified range results in a mismatch. By making FLT_DLC_LO = FLT_DLC_HI, only payloads of the specified quantity of bytes will be filtered. DLC is not maskable.

When the inside range of payloads option is selected (CTRL1_PN[PLFS] = 11), both PL1 and PL2 are considered with the 8-byte data length. All the data bytes excluded by the DLC of the received message are considered with value zero.

Payload mask is only used in the exact match option (CTRL1_PN[PLFS] = 00) to select which bits or bytes in the 8-byte data field of both Rx incoming message and the contents of PL1 register are selected for matching. Mask length must be in accordance with the expected range of DLC values. For maximum range, minimum range, and inside range comparisons, the payload mask is not considered.

When a remote frame is received by FLEXCAN and CTRL1[FCS] is configured to select the payload comparison, the payload filtering is not considered and the comparison results in a mismatch.

Rx incoming messages can also be filtered based upon the quantity and rate of message reception, specifically:

- Several messages that match the filtering criteria for ID or payload for a predefined number of times. This number can be configured in the 1 to 255 range. See the Pretended Networking Control 1 Register (CTRL1_PN).
- No message matching the filtering criteria for ID or payload up to a timeout trigger. That is, non-reception of a matching message for a defined quantity of time. See the Pretended Networking Control 2 Register (CTRL2_PN).

FLEXCAN can generate a wake-up timeout event from an internal timer with associated comparator circuitry capable of generating a timeout flag when the counting reaches the pre-defined timeout value, as specified in CTRL2_PN[MATCHTO].

The above filtering criteria can be used together as follows:

- Message ID filtering only
- Message ID filtering and Payload filtering
- Message ID filtering only occurring N times
- Message ID filtering and Payload filtering occurring N times

The timeout counter runs concurrently with the reception filtering process. Both engines, timeout counter and message filtering, are independent. If an incoming message matches the selected filter criteria, the timeout counter keeps counting until the CPU wakes up.

Conversely, if the timeout counter reaches the target value, then the message filtering process continues to filter incoming messages until the CPU wakes up.

WU_MTC[MCOUNTER] will report the number of matching messages that occurred under Pretended Networking mode up to the moment the CPU wakes up.

Under Pretended Networking mode, the wakeup event that may occur will set the respective wakeup flag (see Pretended Networking Wake Up Match Register (WU_MTC)):

- In case of a successful match in accordance with the selected filtering criteria, WU_MTC[WUMF].
- In case of a timeout trigger, WU_MTC[WTOF].

Any of these flags will generate interrupts to the CPU, provided the respective mask bits are enabled (CTRL1_PN[WUMF_MSK] or CTRL1_PN[WTOF_MSK]).

There are four Wake up Message Buffers (WMBs) used to store incoming messages in Pretended Networking mode. Up to four messages can be stored (see Wake Up Message Buffer register for C/S (WMBCSn)).

When CTRL1_PN[NMATCH] value is one, only one message is received if the filtering criteria are matched, and this message is stored in WMB0.

If NMATCH value is between two and four, WMB1, WMB2, and WMB3 are used to store the second, third, and fourth matching messages respectively.

If NMATCH is greater than four, the last four matching messages are stored in the WMBs. The WMB index indicates the arrival order. The last message is stored in WMB3.

Only the valid data bytes of the incoming match message are stored in the data field of WMBs. The non-valid data bytes are read as zero. If DLC=0 and RTR=1, the data field is filled with zeroes. In any of the above cases, the wakeup interrupt is generated just when the filtering criteria is completed and CTRL1_PN[WUMF_MSK] is enabled.

When a non-match wakeup event occurs (timeout or external) and the MCOUNTER register is equal to or greater than four, the message stored in WMB0 does not have valid content. WMB0 is used as a buffer for the current message in the FLEXCAN bus. Messages received during Pretended Networking mode don't have time stamps, and the respective field in the WMB structure must be ignored.

Under low power mode (Stop), all processes are shut down except for the PN functionality inside the FLEXCAN_PE sub-block, which is kept clocked by the oscillator clock. FLEXCAN continues to receive Rx incoming messages but only compares them to the predefined target values and in accordance with the selected filtering criteria. The matching, arbitration, move-in and move-out processes, normally available in Normal mode, are not performed under Pretended Networking mode.

FLEXCAN under Pretended Networking reacts to messages on the FLEXCAN bus in the same manner as in Normal mode (that is, it generates acknowledge bits, detect and count errors, etc.).

### 60.6.8 Move Process

There are two types of move process: move-in and move-out.

#### *60.6.8.1 Move-in*

The move-in process is the copy of a message received by an Rx SMB to an Rx mailbox or FIFO that has matched it. If the move destination is the Rx FIFO, attributes of the message are also copied to the FLEXCAN_RXFIR FIFO. Each Rx SMB has its own move-in process, but only one is performed at a given time as described ahead. The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (see Matching process) and all of the following conditions are true:

- The FLEXCAN bus has either reached or already gone past:
  - The second bit of Intermission field next to the frame that carried the message that is in the Rx SMB.
  - The first bit of an overload frame next to the frame that carried the message that is in the Rx SMB.
- There is no ongoing matching process.
- The destination mailbox is not locked by the CPU.
- There is no ongoing move-in process from another Rx SMB. If more than one move-in process is to be started at the same time, both are performed and the newest substitutes for the oldest.

The term *pending move-in* is used throughout the documentation and stands for a move-to-be that still does not satisfy all of the aforementioned conditions.

The move-in is cancelled and the Rx SMB is able to receive another message if any of the following conditions is satisfied:

- The destination mailbox is inactivated after the FLEXCAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished.
- There is a previous pending move-in to the same destination mailbox.
- The Rx SMB is receiving a frame transmitted by the FLEXCAN itself and self- reception is disabled (MCR[SRXDIS] is asserted).
- Any CAN protocol error is detected.

Note that the pending move-in is not cancelled if the module enters Freeze or Low-Power mode. It stays on hold, only waiting for Freeze and Low-Power mode to be exited and the module to be unlocked. If an MB is unlocked during Freeze mode, the move-in happens immediately.

The move-in process is the execution by the FLEXCAN of the following steps:

1) Push IDHIT into the RXFIR FIFO if the message is destined for the Rx FIFO.
2) Read all data words from the Rx SMB in accordance with the selected payload size for the Rx storage element.
3) Write all data words to the Rx mailbox in accordance with the selected payload size for the Rx storage element. If the data size of the storage element is smaller than the original payload size described in the message's DLC field, the payload is truncated and the high order bytes that do not fit the destination size are lost.
4) Read the Control/Status and ID words from the Rx SMB.
5) Write Control/Status and ID words to the Rx mailbox, and update the CODE field.

The move-in process is not atomic, in such a way that it is immediately cancelled by the inactivation of the destination mailbox (see Mailbox inactivation). In this case, the mailbox may remain partially updated, thus incoherent. The exception is if the move-in destination is an Rx FIFO message buffer; then the process cannot be cancelled.

The BUSY Bit (least significant bit of the CODE field) of the destination message buffer is asserted while the move-in is being performed to alert the CPU that the message buffer content is temporarily incoherent.

#### *60.6.8.2 Move-out*

The move-out process is the copy of the content from a Tx Mailbox to the Tx SMB when a message for transmission is available (see Arbitration process). The move-out occurs in the following conditions:

- The first bit of Intermission field
- During Bus Off state when TX Error Counter is in the 124 to 128 range
- During Bus Idle state
- During Wait For Bus Idle state

The move-out process is not atomic. Only the CPU has priority to access the memory concurrently out of Bus Idle state. In Bus Idle, the move-out has the lowest priority to the concurrent memory accesses.

### 60.6.9 Data Coherence

To maintain data coherency and FLEXCAN proper operation, the CPU must obey the rules described in Transmit process and Receive process.

#### *60.6.9.1 Transmission Abort Mechanism*

The abort mechanism provides a safe way to request the abort of a pending transmission. A feedback mechanism is provided to inform the CPU if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

Two primary conditions must be fulfilled to abort a transmission:

- MCR[AEN] must be asserted.
- The first CPU action must be the writing of abort code (0b1001) into the CODE field of the Control and Status word.

Active MBs configured for transmission must be aborted first before they can be updated. If the abort code is written to a Mailbox that is currently being transmitted or to a Mailbox that was already loaded into the Tx SMB for transmission, the write operation is blocked and the transmission is not disturbed. However, the abort request is captured and kept pending until one of the following conditions is satisfied:

- The module loses the bus arbitration.

- There is an error during the transmission.
- The module is put into Freeze mode.
- The module enters Bus Off state.
- There is an overload frame.

If none of the conditions above are reached:

- The MB is transmitted correctly.
- The interrupt flag is set in the IFLAG register.
- An interrupt to the CPU is generated (if enabled).

The abort request is automatically cleared when the interrupt flag is set. On the other hand, if only one of the above conditions is reached, the frame is not transmitted; therefore:

- The abort code is written into the CODE field.
- The interrupt flag is set in the IFLAG.
- An interrupt is (optionally) generated to the CPU.

If the CPU writes the abort code before the transmission begins internally, then the write operation is not blocked; therefore, the MB is updated and the interrupt flag is set. In this way, the CPU only needs to read the abort code to make sure the active MB was safely inactivated. Although the AEN bit is asserted and the CPU wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet.

One Mailbox is aborted only when the abort request is captured and kept pending until one of the previous conditions is satisfied.

#### *60.6.9.2 Mailbox Inactivation*

Inactivation is a mechanism provided to protect the mailbox against updates by the FLEXCAN internal processes, thus allowing the CPU to rely on mailbox data coherence after having updated it, even in Normal mode.

Inactivation of transmission mailboxes must be performed just when MCR[AEN] is deasserted.

If a mailbox is inactivated, it participates in neither the arbitration process nor the matching process until it is reactivated. See Transmit process and Receive process for more detailed instructions on how to inactivate and reactivate a mailbox.

To inactivate a mailbox, the CPU must update its CODE field to INACTIVE (either 0b0000 or 0b1000).

Because you will not be able to synchronize the CODE field update with the FLEXCAN internal processes, an inactivation can have the following consequences:

- A frame in the bus that matches the filtering of the inactivated Rx mailbox may be lost without notice, even if there are other mailboxes with the same filter.
- A frame containing the message within the inactivated Tx mailbox may be transmitted without setting the respective IFLAG.

To perform a safe inactivation and avoid the above consequences for Tx mailboxes, the CPU must use the transmission abort mechanism (see Transmission abort mechanism).

The inactivation automatically unlocks the mailbox (see Mailbox lock mechanism).

***Note:*** Message buffers that are part of the Rx FIFO cannot be inactivated. There is no write protection on the FIFO region by FLEXCAN. CPU must maintain data coherency in the FIFO region when RFEN is asserted.

#### *60.6.9.3 Mailbox Lock Mechanism*

Other than mailbox inactivation, FLEXCAN has another data coherence mechanism for the receive process. When the CPU reads the Control and Status word of an Rx MB with codes FULL or OVERRUN, FLEXCAN assumes that the CPU wants to read the whole MB in an atomic operation, and therefore it sets an internal lock flag for that MB. The lock is released when the CPU reads the free-running timer (global unlock operation), or when it reads the Control and Status word of another MB regardless of its code. A CPU write into the C/S word also unlocks the MB, but this procedure is not recommended for normal unlock use because it cancels a pending move and potentially may lose a received message. The MB locking prevents a new frame from being written into the MB while the CPU is reading it.

***Note:*** The locking mechanism applies only to Rx MBs that are not part of the FIFO and have a code different than INACTIVE(0b0000) or EMPTY1 (0b0100). Also, Tx MBs cannot be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FLEXCAN has already received and stored messages into these two MBs. Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no free-to-receive MBs, so it decides to override MB number 5. However, this MB is locked, so the new message cannot be written there. It will remain in the Rx SMB waiting for the MB to be unlocked, and only then will be written to the MB.

If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the Rx SMB and there will be no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved in from the Rx SMB to the MB, the BUSY bit on the CODE field is asserted. If the CPU reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

***Note:*** If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Inactivation takes precedence over locking. If the CPU inactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the Rx SMB will not be transferred anymore to the MB. An MB is unlocked when the CPU reads the Free-Running Timer Register (see Free Running Timer (FLEXCAN_TIMER)), or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both Normal and Freeze modes.

An unlock during Normal or Freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during a low power mode (see Modes of operation), and it takes place only when the module returns to Normal or Freeze modes.

### 60.6.10 Enhanced Rx FIFO

FLEXCAN supports an enhanced Rx FIFO engine which can store up to 12 CAN FD messages. The region 0x2000–0x204F contains the output of the FIFO which should be read by the CPU. The enhanced Rx FIFO is enabled by setting ERFCR[ERFEN].

Although FLEXCAN has two FIFO options, Legacy Rx FIFO and Enhanced Rx FIFO, they cannot be enabled at the same time. Refer to Legacy Rx FIFO to find detailed information about Legacy Rx FIFO.

The enhanced Rx FIFO has a watermark that is configured by setting ERFCR[ERFWM]. If ERFCR[ERFWM] is set, the CPU is notified only if a minimum number of messages is stored in the FIFO. When the number of stored messages is greater than the value in ERFCR[ERFWM], EFFSR[ERFWMI] is set by the hardware. Optionally, an interrupt or a DMA transfer can be triggered if MCR[DMA] or ERFIER[ERFWMIIE] are set respectively.

For the Enhanced Rx FIFO to receive, the CPU must execute the configuration procedure below. The same procedure must be done if the CPU needs to change any of the configuration of the Enhanced RX FIFO.

1) Enter Freeze mode.
2) Enable Enhanced Rx FIFO by setting ERFCR[ERFEN], if it is not already enabled (MCR[RFEN] must be cleared so that Enhanced Rx FIFO is enabled.).
3) Write one to ERFSR[ERFCLR] to reset Enhanced Rx FIFO engine.
4) Clear ERFSR[ERFUFW], ERFSR[ERFOVF], ERFSR[ERFWMI], and ERFSR[ERFDA], if they are set.
5) Write ERFCR[NFE] to configure the total number of enhanced Rx FIFO filter elements to be used in Enhanced Rx FIFO reception.
6) Write ERFCR[NEXIF] to configure the number of extended ID and standard ID filter elements to be used in Enhanced Rx FIFO reception (NEXIF ⩽ NFE + 1).
7) Configure the Enhanced Rx FIFO watermark by writing ERFCR[ERFWM].
8) If interrupts will be used, set the interrupt enables in the ERIER register.
9) If DMA will be used, set MCR[DMA] to enable DMA operation and write ERFCR[DMALW] to configure the number of words to transfer for each Enhanced Rx FIFO data element.
10) Configure the filter elements by writing in the ERFFELn registers (ERFFELn registers are implemented in RAM; thus, they must be explicitly initialized prior to any reception).
11) Leave Freeze mode.

There are two types of Enhanced Rx FIFO filter elements that can be stored in ERFFELn registers: extended-ID filter element and standard-ID filter element. Each extended-ID filter element is stored in two ERFFELn registers while each standard-ID filter element is stored in one ERFFELn register. The total number of Enhanced RX FIFO filter elements is defined by ERFCR[NFE].

In addition, the filter memory space can be split into two regions: one for extended-ID filter elements and another for standard-ID filter elements, according to ERFCR[NEXIF]. The figure below shows how the Enhanced Rx filter elements are defined. Detailed information about the Enhanced Rx FIFO matching process and filter element formats can be found in the Enhanced Rx FIFO matching process.

#### *60.6.10.1 Enhanced Rx FIFO Matching Process*

When ERFCR[ERFEN] is enabled, FLEXCAN scans the ERFFELn memory region. If at least one filter element satisfies the matching criteria, then the CAN message content is transfered to the enhanced RX FIFO memory.

Each ERFFELn register can store one standard filter element. The matching criteria are defined by ERFFELn[FSCH] in this way:

- If FSCH = b00, the filter scheme is based on mask and filter. A CAN message matches a standard ID filter element only if these criteria are reached:
  a. CAN message is base frame format (IDE = 0)
  b. (ID[n] = STD ID filter [n]) or (STD ID Mask[n] =0) for each bit n from 0 to 10
  c. (RTR = RTR Filter) or (RTR MASK = 0)

  In this explanation, RTR and ID are the Remote Transmit Request field and the ID from a CAN message, respectively.

  If FSCH = b00, the filters and masks are defined as shown in the following table.

**Table 95. Standard ID filter element with filter + mask scheme (FSCH = b00)**

| 31-30 | 29-28 | 27 | 26-16 | 15-12 | 11 | 10-0 |
|---|---|---|---|---|---|---|
| FSCH=b00 | Reserved | RTR filter | STD ID filter | Reserved | RTR mask | STD ID mask |

- If FSCH = b01, the filter scheme is based on range. A CAN message matches a standard ID filter element only if these criteria are reached:
  a. CAN message is base frame format (IDE = 0)
  b. ID ⩾ STD ID Filter1
  c. ID ⩽ STD ID Filter2
  d. (RTR = RTR filter) or (RTR MASK = 0)

  RTR and ID are the Remote Transmit Request bit and ID from a CAN message, respectively. If FSCH = b01, the filters and mask are defined as shown in the following table.

**Table 96. Standard ID filter element with range scheme (FSCH = b01)**

| 31-30 | 29-28 | 27 | 26-16 | 15-12 | 11 | 10-0 |
|---|---|---|---|---|---|---|
| FSCH=b01 | Reserved | RTR filter | STD ID Filter2 | Reserved | RTR mask | STD ID Filter1 |

- If FSCH = b10, the filter scheme is based on two filters without masks. A CAN message matches a standard ID filter element only if these criteria are reached:
  a. CAN message is base frame format (IDE = 0)
  b. (ID[n] = STD ID Filter1[n]) or (ID[n] = STD ID Filter2[n]) for each bit n from 0 to 10
  c. (RTR = RTR Filter1) or (RTR = RTR Filter2)

  RTR and ID are the Remote Transmit Request bit and ID from a CAN message, respectively. If FSCH = b10, the filters are defined as shown in the following table.

**Table 97. Standard ID filter element with two-filter scheme (FSCH = b10)**

| 31-30 | 29-28 | 27 | 26-16 | 15-12 | 11 | 10-0 |
|---|---|---|---|---|---|---|
| FSCH=b10 | Reserved | RTR Filter2 | STD ID Filter2 | Reserved | RTR Filter1 | STD ID Filter1 |

Each pair of ERFFELn registers can store one extended filter element. The matching criteria are defined by ERFFELn[FSCH] in this way:

- If FSCH = b00, the filter scheme is based on mask and filter. A CAN message matches an extended ID filter element only if these criteria are reached:
  a. CAN message is extended frame format (IDE = 1)
  b. (ID[n] = EXT ID filter [n]) or (EXT ID Mask[n] =0) for each bit n from 0 to 28
  c. (RTR = RTR Filter) or (RTR MASK = 0)

  If FSCH = b00, the filters and masks are defined as shown in the following table.

**Table 98. Extended ID filter element with filter + mask scheme (FSCH = b00)**

| 31-30 | 29 | 28-0 |
|---|---|---|
| FSCH | RTR filter | EXT ID filter |
| Reserved | RTR mask | EXT ID mask |

- If FSCH = b01, the filter scheme is based on range. A CAN message matches an extended ID filter element only if the following criteria are reached:
  a. CAN message is extended frame format (IDE = 1)
  b. ID ⩾ EXT ID Filter1
  c. ID ⩽ EXT ID Filter2
  d. (RTR = RTR Filter) or (RTR MASK = 0)

If FSCH = b01, the filters and masks are defined as shown in the following table.

**Table 99. Extended ID filter element with range scheme (FSCH = b01)**

| 31-30 | 29 | 28-0 |
|---|---|---|
| FSCH | RTR filter | EXT ID Filter2 |
| Reserved | RTR mask | EXT ID filter 1 |

- If FSCH = b10, the filter scheme is based on two filters without masks. A CAN message matches an extended ID filter element only if these criteria are reached:
  a. CAN message is extended frame format (IDE = 1)
  b. (ID[n] = EXT ID Filter1[n]) or (ID[n] = EXT ID Filter2[n]) for each bit n from 0 to 28
  c. (RTR = RTR Filter1) or (RTR = RTR Filter2)

If FSCH = b10, the filters and masks are defined as shown in the following table.

**Table 100. Extended ID filter element with two-filter scheme (FSCH = b10)**

| 31-30 | 29 | 28-0 |
|---|---|---|
| FSCH | RTR Filter2 | EXT ID Filter2 |
| Reserved | RTR Filter1 | EXT ID filter 1 |

### *60.6.10.2 Enhanced Rx FIFO under DMA operation*

***Note:*** The DMA controller must perform the number of read loops in the FIFO output defined by ERFCR[ERFWM]. Also, each message reading process must end by the address defined in ERFCR[DMALW].

The DMA feature is enabled by asserting both ERF[ERFEN] and MCR[DMA]. The DMA controller can read the received message by reading a message buffer structure at the Enhanced FIFO output port at the address range defined in the Enhanced Rx FIFO structure.

The CPU should not access the Enhanced FIFO output port address range during DMA operation, so that the FIFO engine operates properly. Before enabling MCR[DMA], the CPU must service Enhanced Rx FIFO status bits. Otherwise, these bits may show that the FIFO has data to be serviced, and mistakenly generate a DMA request. Before disabling MCR[DMA], the CPU must first clear the ERFUFW, ERFOVF, ERMWMI, and ERFDA fields in the ERFSR register, and then clear the Enhanced RX FIFO engine by writing one to ERFSR[ERFCLR].

ERFSR[ERFDA] is set by the hardware when there is at least one frame available to be read from the Enhanced Rx FIFO, and ERFSR[ERFWMI] is set by the hardware when the number of frames stored in the FIFO is greater than the watermark defined by ERFCR[ERFWM]. A DMA request is generated when ERFSR[ERFWMI] is set. Upon receiving the request, the DMA controller can read the message in the Enhanced Rx FIFO output.

The following rules must be obeyed for Enhanced Rx FIFO DMA operation:

- Because a DMA transfer cannot be dynamically changed, ERFCR[DMALW] should be programmed so that the Enhanced Rx FIFO element can store the largest CAN message present on the FLEXCAN bus.
- Data bytes are valid according to the DLC field.

Each time one message is read from FIFO by the DMA controller, ERFSR[ERFDA] is cleared by FLEXCAN and set again if there is at least one message stored in the FIFO. Moreover, ERFSR[ERFWMI] is cleared by the hardware when the messages are read in the FIFO output, and set again if the number of messages in the FIFO is greater than the value defined by ERFCR[ERFWM].

For example, suppose that the maximum number of bytes in the data field of a CAN frame for a certain application is eight, and a high-resolution time stamp is enabled. In that case, the last Enhanced RX FIFO address offset can be found in the Enhanced Rx (ERX) FIFO structure. Using this address offset, ERFCR[DMALW] can be determined like this:

- Maximum number of data bytes = 8
- Last address offset = ID_HIT = 0x2010
- DMALW = 4

#### *60.6.10.3 Enhanced Rx FIFO Clear Operation*

When ERFCR[ERFEN] is set, the CPU can clear the Enhanced Rx FIFO by writing one into ERFSR[ERFCLR]. The clear operation resets the internal FIFO pointers although the FIFO content stored in RAM is not changed. This operation can only be performed in Freeze mode and is blocked by hardware in other modes. This operation does not clear the ERFUFW, ERFOVF, ERFDA, and ERFWMI fields in ERFSR. Consequently, the CPU must service all these fields before executing the clear FIFO task.

### 60.6.11 Legacy RX FIFO

The Rx FIFO is receive-only and is enabled by asserting MCR[RFEN]. The reset value of this bit is zero to maintain software backward compatibility with previous versions of the module that did not have the FIFO feature.

***Note:*** Rx FIFO must not be enabled when the FLEXCAN FD feature is enabled.

The FIFO is 6-message deep. The memory region is occupied by the FIFO structure (both message buffers and FIFO engine). The CPU can read the received messages sequentially, in the order they were received, by repeatedly reading a message buffer structure at the output of the FIFO.

IFLAG1[BIT5] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, the CPU can read the message (accessing the output of the FIFO as a message buffer) and the RXFIR register, and then clear the interrupt. If there are more messages in the FIFO the act of clearing the interrupt updates the output of the FIFO with the next message and updates RXFIR with the attributes of that message, reissuing the interrupt to the CPU. Otherwise, the flag remains negated. The output of the FIFO is valid only while IFLAG1[BIT5] is asserted.

IFLAG1[BIT6] (Rx FIFO Warning) is asserted when the number of unread messages within the Rx FIFO is increased to five from four due to the reception of a new one, meaning that the Rx FIFO is almost full. The flag remains asserted until the CPU clears it.

IFLAG1[BIT7] (Rx FIFO Overflow) is asserted when an incoming message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. The flag remains asserted until the CPU clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated if an IFLAG bit is asserted and the corresponding mask bit is asserted too.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing workload. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to CTRL2[RFFN] setting, that can be configured to one of the following formats:

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

***Note:*** A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

Every frame available in the FIFO has a corresponding IDHIT (Identifier Acceptance Filter Hit Indicator) that can be read in the IDHIT field from C/S word, as shown in the Rx FIFO Structure description. Another way the CPU can obtain this information is by accessing the RXFIR register. RXFIR[IDHIT] refers to the message at the output of the FIFO and is valid while the IFLAG1[BIT5] flag is asserted. The RXFIR register must be read-only before clearing the flag, which guarantees that the information refers to the correct frame within the FIFO.

Up to 32 elements of the filter table are individually affected by the Individual Mask Registers (RXIMRx), according to the setting of CTRL2[RFFN], allowing very powerful filtering criteria to be defined. If MCR[IRMQ] is negated, then the FIFO filter table is affected by RXFGMASK.

#### *60.6.11.1 Rx FIFO under DMA Operation*

The receive-only FIFO can support DMA. This feature is enabled by asserting both MCR[RFEN] and MCR[DMA]. The reset value of MCR[DMA] is zero to maintain backward compatibility with previous versions of the module that did not have the DMA feature.

The DMA controller can read the received message by reading a message buffer structure at the FIFO output port at the 0x80-0x8C address range.

When MCR[DMA] is asserted the CPU must not access the FIFO output port address range. Before enabling MCR[DMA], the CPU must service the IFLAGs asserted in the Rx FIFO region. Otherwise, these IFLAGs may show that the FIFO has data to be serviced, and mistakenly generate a DMA request. Before disabling MCR[DMA], the CPU must perform a clear FIFO operation.

IFLAG1[BIT5] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO. Consequently, a DMA request is generated simultaneously. Upon receiving the request, the DMA controller can read the message (accessing the output of the FIFO as a message buffer). The DMA reading process must end by reading address 0x8C, which clears IFLAG1[BIT5] and updates both the FIFO output with the next message (if FIFO is not empty) and the RXFIR register with the attributes of the new message. If there are more messages stored in the FIFO, IFLAG1[BIT5] will be re-asserted and another DMA request is issued. Otherwise, the flag remains negated.

***Note:*** RXFIR register contents cannot be read after DMA completes the FIFO read. The IDHIT information is also available in the C/S word at address 0x080 (see Rx FIFO structure.

IFLAG1[BIT6] and IFLAG1[BIT7] are not used when the DMA feature is enabled.

When FLEXCAN is working with DMA, the CPU does not receive any Rx FIFO interruption and must not clear the related IFLAGs. In addition, the related IMASKs are not used to mask the generation of DMA requests.

#### *60.6.11.2 Clear FIFO Operation*

When MCR[RFEN] is asserted, the clear FIFO operation is a feature used to empty FIFO contents. With MCR[RFEN] asserted the Clear FIFO occurs when the CPU writes one in IFLAG1[BUF0I]. This operation can only be performed in Freeze mode and is blocked by hardware in other modes. This operation does not clear the FIFO IFLAGs; consequently, the CPU must service all FIFO IFLAGs before executing the clear FIFO task.

When Rx FIFO is working with DMA, the clear FIFO operation clears IFLAG1[BIT5] and the DMA request is canceled.

***Note:*** Clear FIFO operation does not clear IFLAGs, except when MCR[DMA] is asserted; in this case, only IFLAG1[BIT5] is cleared.

### 60.6.12 CAN Protocol Related Features

#### *60.6.12.1 CAN FD ISO Compliance*

The CAN FD protocol has been improved to increase the failure detection capability that was in the original CAN FD protocol, which is also called non-ISO CAN FD, by CAN in Automation (CiA). A three-bit stuff counter and a parity bit have been introduced in the improved CAN FD protocol, now called ISO CAN FD. The CRC calculation has also been modified. All these improvements make the ISO CAN FD protocol incompatible with the non-FD CAN FD protocol. The non-ISO CAN FD is still supported by FLEXCAN so that it can be used mainly during an intermediate phase, for evaluation and development purposes.

Therefore, it is strongly recommended to configure FLEXCAN to the ISO CAN FD protocol by setting the ISOCANFDEN field in the CTRL2 register.

#### *60.6.12.2 Transceiver Delay Compensation*

The CAN FD protocol allows the transmission and reception of data at a higher bit rate than the nominal rate used in the arbitration phase when the message's BRS bit is set. This feature enables the use of rates up to 8 Mbps.

During the data phase of a CAN FD frame, the transmitter detects a bit error if it cannot receive its own latest transmitted bit at the sample point of that bit. When bit rate switching is enabled (BRS bit is asserted), the length of the CAN bit time in the data phase can become shorter than the transceiver's loop delay, thus impeding the correct comparison between the transmitted bit and the received bit within the current CAN bit time interval.

The FLEXCAN supports an optional Transceiver Delay Compensation (TDC) mechanism that defines a secondary sample point where the transmitted bit is correctly compared with the received bit in order to check for bit errors.

The TDC mechanism is enabled by the FLEXCAN_FDCTRL[TDCEN] bit and is effective only during the data phase of CAN FD frames having the BRS bit set. It has no effect either on non-FD frames, or on FD frames transmitted at the normal bit rate. The TDC is active from the sample point of the BRS bit until the sample point of the CRC Delimiter bit, provided the respective message under transmission has the BRS bit set. When it is active, a comparison is done between the real received bit and the delayed transmitted bit, where the delay is calculated based on the measured transceiver loop delay.

For every transmitted FD frame having the BRS bit set, the delay measurement is triggered by the transition from the recessive EDL bit to the dominant R0 bit.

The loop delay is measured in PE clock periods, from the transmitted EDL-R0 edge to the received EDL-R0 edge. The position of the secondary sample point is defined by the measured loop delay time added to an offset value specified in

FLEXCAN_FDCTRL[TDCOFF]. The FLEXCAN_FDCTRL[TDCVAL] bit field stores the result of this calculation. The TDCVAL value saturates at its maximum value of 15 CANCLK when the delay measurement is too long.

The measured loop delay is not enough to be used to define the secondary sample point because it relates to the CAN bit edges. The transceiver delay compensation offset TDCOFF is used to shift the secondary sample point from the edge to an intermediate point inside the bit time (for example, half of the bit time in the data phase), far away from its edges. Therefore, the TDCOFF value cannot be larger than the CAN bit duration in the data phase.

During the data phase of CAN FD frames with bit rate switching enabled, at the onset of every Tx CAN bit, the transmitted Tx bit value is temporarily stored in a buffer and a time countdown based on TDCVAL is started which ends with the comparison of the received Rx bit (delayed by the external loop delay plus the specified offset) with the stored Tx bit. If a bit error is detected at the secondary sample point, the FLEXCAN issues an error flag to the FLEXCAN bus at the next sample point.

During the arbitration phase, the delay compensation is always disabled. The maximum delay that can be compensated by the FLEXCAN's transceiver delay compensation during the data phase is 3 CAN bit times - 2 Tq. Beyond this limit, the FLEXCAN_FDCTRL[TDCFAIL] flag is set to indicate when the Transceiver Delay Compensation mechanism is out of range and unable to compensate for the transceiver loop delay.

#### *60.6.12.3 Protocol Timing*

The FLEXCAN module supports a variety of means to set up bit timing parameters that are required by the CAN protocol. The Control 1 register (CTRL1) has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW.

The CAN Bit Timing register (CBT) extends the range of the CAN bit timing variables in CTRL1. The CAN FD Bit Timing register (FDCBT) provides a second set of CAN bit timing variables to be applied at the data phase of CAN FD frames with the Bit Rate Switch (BRS) set.

The Enhanced Nominal CAN Bit Timing register (ENCBT) extends the range of CAN bit timing variables in CBT. The Enhanced Data Phase CAN Bit Timing register (EDCBT) extends the range of CAN bit timing variables in FDCBT. When using ENCBT and EDCBT, the bit timing nominal and data phase serial clock (Sclock) dividers must be programed in the Enhanced CAN Bit Timing Prescalers register (EPRS).

The PRESDIV field (as well as its extended range EPRESDIV (or ENPRESDIV) and FPRESDIV (or EDPRESDIV) for the data phase bits of CAN FD messages) defines the prescaler value (see the equation below) that generates the serial clock (Sclock), whose period defines the time quantum used to compose the FLEXCAN waveform. A time quantum (Tq) is the atomic unit of time managed by the FLEXCAN engine.

$$T_q = \frac{PRESDIV + 1}{f_{canclk}}$$

The bit rate, which defines the rate the CAN message is either received or transmitted, is given by the formula:

$$CAN\ Bit\ Time = (Number\ of\ Time\ Quanta\ in\ 1\ bit\ time) * T_q$$

$$Bit\ Rate = \frac{1}{CAN\ Bit\ Time}$$

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time Segment 1: This segment includes the propagation segment and the phase segment 1 of the FLEXCAN standard. It can be programmed by setting CTRL1[PROPSEG] and CTRL1[PSEG1] so that their sum (plus 2) is in the range of 2 to 16 time quanta. When CBT[BTF] is asserted, FLEXCAN uses EPROPSEG and EPSEG1 fields from the CBT register so that their sum (plus 2) is in the range of 2 to 96 time quanta. For messages in CAN FD format with the

BRS bit set, FLEXCAN uses FPROPSEG and FPSEG1 from FDCBT instead, so that their sum (plus 1) is in the range of 2 to 39 time quanta.

If CTRL2[BTE] is set, FLEXCAN uses ENCBT[NTSEG1] to configure time segment 1 in the range of 2 to 256 time quanta. For the data phase in CAN FD messages with BRS set, EDCBT[DTSEG1] must be used for configuring time segment 1 in the range of 2 to 32 time quanta.

- Time Segment 2: This segment represents the phase segment 2 of the FLEXCAN standard. It can be programmed by setting CTRL1[PSEG2] (plus 1) to be 2 to 8 time quanta long. When CBT[BTF] is asserted, FLEXCAN uses EPSEG2 fields of CBT register so that its value (plus 1) is in the range of 2 to 32 time quanta. For messages in CAN FD format with the BRS bit set, FLEXCAN uses FPSEG2 from FDCBT instead, so that its value (plus 1) is in the range of 2 to 8 time quanta. The time segment 2 cannot be smaller than the Information Processing Time (IPT), which value is 2 time quanta in FLEXCAN.

If CTRL2[BTE] is set, FLEXCAN uses ENCBT[NTSEG2] to configure time segment 2 in the range of 2 to 128 time quanta. For the data phase in CAN FD messages with BRS set, EDCBT[DTSEG2] must be used for configuring time segment 2 in the range of 2 to 16 time quanta.

***Note:*** The bit time defined by the above time segments must not be smaller than five time quanta. For bit time calculations, use an Information Processing Time (IPT) of two, which is the value implemented in the FLEXCAN module.

**Table 101. Time segment syntax**

| Syntax | Description |
|---|---|
| SYNC_SEG | System expects transitions to occur on the bus during this period. |
| TSEG1 | Corresponds to the sum of PROPSEG and PSEG1. |
| TSEG2 | Corresponds to the PSEG2 value. |
| Transmit point | A node in transmit mode transfers a new value to the FLEXCAN bus at this point. |
| Sample point | A node samples the bus at this point. If the three samples per bit option are selected, then this point marks the position of the third sample. |

The following table gives some examples of the FLEXCAN compliant segment settings for Classical CAN format (Bosch CAN 2.0B) (non-FD) messages.

**Table 102. Bosch CAN 2.0B standard compliant bit time segment settings**

| Time Segment 1 | Time Segment 2 | Re-synchronization Jump Width |
|---|---|---|
| 5 .. 10 | 2 | 1 .. 2 |
| 4 .. 11 | 3 | 1 .. 3 |
| 5 .. 12 | 4 | 1 .. 4 |
| 6 .. 13 | 5 | 1 .. 4 |
| 7 .. 14 | 6 | 1 .. 4 |
| 8 .. 15 | 7 | 1 .. 4 |
| 9 .. 16 | 8 | 1 .. 4 |

Whenever CAN bit is used as a measure of time duration (for example, estimating the occurrence of a CAN bit event in a message), the number of peripheral clocks in one CAN bit (NumClkBit) can be calculated as:

$$NumClkBit = \frac{f_{sys}}{f_{canclk}} * (PRESDIV + 1) * (PROPSEG + PSEG1 + PSEG2 + 4)$$

or, if CTRL2[BTE] = 1:

$$NumClkBit = \frac{f_{sys}}{f_{canclk}} * (ENPRESDIV + 1) * (NTSEG1 + NTSEG2 + 3)$$

where:

- NumClkBit is the number of peripheral clocks in one CAN bit.
- fcanclk is the Protocol Engine (PE) Clock, in Hz.
- fsys is the frequency of operation of the system (CHI) clock, in Hz.
- PSEG1 is the value in CTRL1[PSEG1] field.
- PSEG2 is the value in CTRL1[PSEG2] field.
- PROPSEG is the value in CTRL1[PROPSEG] field.
- PRESDIV is the value in CTRL1[PRESDIV] field.
- ENPRESDIV is the value in EPRS[ENPRESDIV] field.
- NTSEG1 is the value in ENCBT[NTSEG1] field.
- NTSEG2 is the value in ENCBT[NTSEG2] field.

The formula above is also applicable to the alternative CAN bit timing variables described in the CAN Bit Timing register (CBT) or the Enhanced Nominal CAN Bit Timing register (ENCBT) and also to the CAN FD Bit Timing register (FDCBT) or the Enhanced Nominal CAN Bit Timing register (ENCBT).

For example, 180 CAN bits = (180 x NumClkBit) peripheral clock periods.

### 60.6.13 Modes of Operation Details

#### *60.6.13.1 Freeze Mode*

This mode is requested either by the CPU through the assertion of MCR[HALT] or when the chip is put into Debug mode. In both cases, it is also necessary that MCR[FRZ] must be asserted and the module is not in a low-power mode.

The acknowledgement is obtained through the assertion by the FLEXCAN of MECR[FRZ_ACK]. The CPU must only consider the FLEXCAN in Freeze mode when both request and acknowledgement conditions are satisfied.

When Freeze mode is requested, FLEXCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off, or Idle state.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in does not prevent entering Freeze mode
- Ignores the Rx input pin and drives the Tx pin as recessive.
- Stops the prescaler, thus halting all CAN protocol activities.
- Grants write access to the Error Counters register, which is read-only in other modes.
- Sets MCR[NOTRDY] and MCR[FRZACK].

After requesting Freeze mode, the user must wait for MCR[FRZ_ACK] to be asserted before executing any other action; otherwise, FLEXCAN may operate unpredictably. In Freeze mode, all memory-mapped registers are accessible, except for CTRL1[CLKSRC], which can be read but cannot be written.

Exiting Freeze mode is done in one of the following ways:

- CPU negates MCR[FRZ].
- The chip is removed from Debug mode and/or the HALT bit is negated.

MCR[FRZ_ACK] is negated after the protocol engine recognizes the negation of the freeze request. When out of Freeze mode, FLEXCAN tries to resynchronize to the FLEXCAN bus by waiting for 11 consecutive recessive bits.

#### *60.6.13.2 Module Disable Mode*

This low-power mode is normally used to temporarily disable a complete FLEXCAN block, with no power consumption. It is requested by the CPU through the assertion of MCR[MDIS], and the acknowledgement is obtained through the assertion by the FLEXCAN of MCR[LPMACK]. The CPU must only consider the FLEXCAN in Disable mode when both request and acknowledgement conditions are satisfied.

If the module is disabled during Freeze mode, it requests to disable the clocks to the PE and CHI submodules, sets MCR[LPMACK], and negates MCR[FRZACK].

It is not recommended to use Module Disable mode under Pretended Networking mode. Negate MCR[MDIS] and wait for MCR[LPMACK] to negate before setting MCR[PNET_EN].

If the module is disabled during transmission or reception, FLEXCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive.
- Shuts down the clocks to the PE and CHI submodules
- Sets MCR[NOTRDY] and MCR[LPMACK].

The Bus Interface Unit continues to operate, enabling the CPU to access memory-mapped registers, except the Rx Mailboxes Global Mask registers, the Rx Buffer 14 Mask register, the Rx Buffer 15 Mask register, the Rx FIFO Global Mask register. The Rx FIFO Information register, the message buffers, the Rx Individual Mask registers, and the reserved words within RAM may not be accessed when the module is in Disable mode. Exiting from this mode is done by negating the MDIS bit by the CPU, which causes the FLEXCAN to request to resume the clocks and negate the LPMACK bit after the CAN protocol engine recognizes the negation of Disable mode requested by the CPU.

#### *60.6.13.3 Stop Mode*

This is a system low-power mode in which all chip clocks can be stopped for maximum power savings. The Stop mode is globally requested by the CPU and acknowledgement is obtained through the assertion by the FLEXCAN of a Stop Acknowledgement signal. The CPU must only consider the FLEXCAN in Stop mode when both request and acknowledgement conditions are satisfied.

If FLEXCAN receives the global Stop mode request during Freeze mode, it sets MCR[LPMACK], negates MCR[FRZACK], and then sends the Stop Acknowledge signal to the CPU, in order to shut down the clocks globally.

If Stop mode is requested during transmission or reception, FLEXCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Sets MCR[NOTRDY] and MCR[LPMACK]

- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Stop mode is exited when the CPU resumes the clocks and removes the Stop mode request.

After the CAN protocol engine recognizes the negation of the Stop mode request, the FLEXCAN negates MCR[LPMACK]. FLEXCAN will then wait for 11 consecutive recessive bits to synchronize to the FLEXCAN bus.

#### *60.6.13.4 Pretended Networking mode*

This is a special low-power mode used to receive wake-up messages with low power consumption. This mode can be selected to operate together with Stop mode. Before entering one of these low-power modes, MCR [PNET_EN] must be asserted. After entering the low-power mode, the CHI subblock is shut down and the FLEXCAN_PE subblock is kept active, so that the Rx receive process is still active to filter incoming messages (see Receive process under Pretended Networking mode) as defined by the configuration registers (see Pretended Networking Control 1 register (FLEXCAN_CTRL1_PN)). Upon detecting a wake-up event, a wake-up interrupt is issued to the system.

To enter Pretended Networking mode, FLEXCAN must be in normal mode (neither in Freeze nor in Disable mode). Under Pretended Networking mode, FLEXCAN keeps itself synchronized with the FLEXCAN BUS in Stop mode. Then, when Stop mode is requested, FLEXCAN performs the following steps:

- Waits to be in Idle state, or else waits for the third bit of Intermission, and then checks it to be recessive.
- Sets MCR[LPM_ACK].
- Requests the shutdown of the CHI submodule clock, while keeping the PE submodule clock active.

FLEXCAN can exit Pretended Networking mode in the following ways:

- The CPU removes the Stop Mode request.
- FLEXCAN will wait until Bus Idle or the third bit of Intermission state to negate MCR[LPM_ACK].

The above exit ways can be triggered either by the FLEXCAN action upon detecting a wakeup event and issuing the respective interrupt, or by the CPU itself upon being waked up by another method. In consequence, FLEXCAN will wait until the Bus Idle state or until the third bit of Intermission state to negate MCR[LPM_ACK] and resume Normal mode. This procedure ensures that FLEXCAN will be synchronized to the FLEXCAN bus after exiting Pretended Networking mode. The CPU must wait for MCR[LPM_ACK] to be negated before performing any access to FLEXCAN.

### 60.6.14 Interrupts

The module has many interrupt sources: interrupts due to message buffers and interrupts due to the ORed interrupts from MBs, Bus Off, Bus Off Done, Error, Error Fast (errors detected in the data phase of CAN FD format messages with the BRS bit set), Wake Up Match, Wake Up Timeout, Tx Warning, and Rx Warning.

Each one of the message buffers can be an interrupt source if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception.

Each of the buffers has an assigned flag bit in the IFLAG registers. The bit is set when the corresponding buffer completes a successful transfer and is cleared when the CPU writes one to it (unless another interrupt is generated at the same time).

***Note:*** It must be guaranteed that the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

If the Rx FIFO is enabled (MCR[RFEN] = 1) and DMA is disabled (MCR[DMA] = 0), the interrupts corresponding to MBs 0 to 7 have different meanings. Bit 7 of the IFLAG1 register becomes the "FIFO Overflow" flag; bit 6 becomes the "FIFO Warning"

flag, bit 5 becomes the "Frames Available in FIFO" flag, and bits 4-0 are unused. See the description of Interrupt Flags 1 register (FLEXCAN_IFLAG1) for more information.

If both Rx FIFO and DMA are enabled (MCR[RFEN] and MCR[DMA] = 1) the FLEXCAN does not generate any FIFO interrupt. Bit 5 of the IFLAG1 register still indicates "Frames Available in FIFO" and generates a DMA request. Bits 7, 6, 4-0 are unused.

***Note:*** FIFO cannot be enabled when the FLEXCAN FD feature is enabled.

For a combined interrupt where multiple MB interrupt sources are OR'd together, the interrupt is generated when any of the associated MBs (or FIFO, if applicable) generates an interrupt. In this case, the CPU must read the IFLAG registers to determine which MB or FIFO source caused the interrupt.

The following interrupt sources generate interrupts like the MB interrupt sources, and can be read from the ESR1 register

- Bus Off
- Bus Off Done
- Error
- Error Fast
- Tx Warning
- Rx Warning

The Bus Off, Error, Tx Warning, and Rx Warning interrupt mask bits are located in the CTRL1 register.

The interrupt sources for Pretended Networking (Wake up by Match Flag and Wake up by Timeout Flag) can be read in the WU_MTC register and the respective interrupt mask bits are located in the CTRL1_PN register.

### 60.6.15 Bus interface

CPU access to FLEXCAN registers is subject to the following rules:

- Unrestricted read and write access to supervisor registers results in an access error.
- Read and write access to implemented reserved address space results in an access error.
- Write access to positions whose bits are all currently read-only results in an access error. If at least one of the bits is not read-only then no access error is issued. Write permission to positions or some of their bits can change depending on the mode of operation or transitory state. See register and field descriptions for details.
- Read and write access to unimplemented address space results in access error.
- Read and write access to RAM located positions during Low Power mode results in an access error.
- It is possible for the RXIMR memory region to be considered as general-purpose memory and available for access. There are two ways of doing this:
  a. If MCR[IRMQ] is cleared, the individual masks (RXIMR) are disabled. In this case, the RXIMR memory region is considered as general-purpose memory.
  b. If MCR[MAXMB] is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general-purpose RAM space. Note that reserved words within RAM cannot be used. As an example, suppose FLEXCAN's RAM can support up to 16 MBs, CTRL2[RFFN] is 0x0, and MCR[MAXMB] is programmed with zero. The maximum number of MBs in this case becomes one. The RAM starts at 0x0080, and the space from 0x0080 to 0x008F is used by the one MB. The memory space from 0x0090 to 0x017F is available. The space between 0x0180 and 0x087F is reserved. The space from 0x0880 to 0x0883 is used by the one individual mask and the available memory in the mask registers space would be from 0x0884 to 0x08BF. From 0x08C0 through 0x09DF there are reserved words for internal use which cannot be used as general-purpose RAM. Generally, free memory space for general purposes depends only on MAXMB.

c. If MCR[FDEN] is enabled, the general-purpose memory can be used out of Freeze mode only.

### 60.6.16 Detection and Correction of Memory Errors

FLEXCAN supports detection and correction of errors in memory read accesses. Each byte of FLEXCAN memory is associated with five parity bits that ensure a Hamming distance of four. The error correction mechanism ensures that in this 13-bit word, errors in one bit can be corrected (correctable errors) and errors in two bits can be detected but not corrected (non-correctable errors). Errors in more than two bits may not be detected. In case of non-correctable errors, the corrupted data is not changed by the error correction logic. When a read access is performed, the parity bits are used to calculate a syndrome, which indicates the error in each byte.

FLEXCAN detects a non-correctable error in the event either an all-zeros or an all-ones read occurs. See description of Error Report Syndrome register (RERRSYNR).

Memory errors are indicated to the host through the status register (Error Status Register (ERRSR)) and bus transfer errors, and reported through report registers (Error Report Address register (RERRAR), Error Report Data register (RERRDR), and Error Report Syndrome register (RERRSYNR)).

The error detection and correction mechanism can be activated or not, controlled by MECR[ECCDIS]. When disabled, updates on indications and reporting registers are stopped, but the parity bits are still calculated and written along with data in memory write operations to ensure that memory has consistent parity bits associated with the data.

***Note:*** All FLEXCAN memory must be initialized before starting its operation in order to have the parity bits in memory properly updated. CTRL2[WRMFRZ] grants write access to all memory positions that require initialization, ranging from 0x080 to 0xADF and from 0xC20 to 0x31FF. The RXMGMASK, RX14MASK, RX15MASK, and RXFGMASK registers need to be initialized as well. MCR[RFEN] and ERF[ERFEN] must not be set during memory initialization.

To avoid accidentally changing the critical error correction configuration, this protocol must be followed to enable the update of the Memory Error Control register (MECR):

1) By default, CTRL2[ECRWRE] is zero and MECR[ECRWRDIS] is one.
2) Set CTRL2[ECRWRE].
3) Clear MECR[ECRWRDIS].
4) All writes to the Memory Error Control register (MECR) must keep MECR[ECRWRDIS] cleared.
5) After configuration is done, lock the Memory Error Control register (MECR) by either setting MECR[ECRWRDIS] or clearing CTRL2[ECRWRE].

#### *60.6.16.1 Sources of the Memory Access*

The FLEXCAN memory can be accessed by two major sources (or requestors):

- Host (CPU): the largest word accessed is 32-bit.
- FLEXCAN internal processes (Rx matching, Tx arbitration, move-in on reception, move-out on transmission): the largest word accessed is 64-bit.

The way that non-correctable errors are indicated and reported depends on the source of access.

#### *60.6.16.2 Error Indication*

Memory errors are indicated by flags HANCEIF, FANCEIF, and CEIF in the Error Status Register (ERRSR). Non-correctable errors detected in memory reads requested by the host are indicated separately than the ones detected in requests by FLEXCAN internal processes. FLEXCAN makes no distinction of the source of the access when correctable errors are detected. There are three independent flags for these three cases, and each flag will raise an interrupt unless it is masked by mask bits

in Memory Error Control Register (MECR). If both non-correctable and correctable errors are found in different bytes in the same read operation, both flags are set.

A non-correctable error detected in host access is also indicated as a bus transfer error. A bus wait request may be asserted to extend the memory transaction to the moment the report registers are updated. This indication cannot be masked. If the flag bit ERRSR[HANCEIF] is not masked, the same non-correctable error will raise a bus transfer error and an interrupt request.

Each indication flag has one overrun flag in Error Status Register (ERRSR). The overrun flags do not request interrupts. Overrun flags for non-correctable errors indicate that other errors of the same nature were detected after the current error being treated; overrun flags for correctable errors indicate that other errors of the same nature were detected before the current error being treated.

This is the recommended handling sequence for error indication:

1) Get error report information from report registers.
2) Use this information to take proper measures in the application.
3) Clear the HANCEIF, FANCEIF, and CEIF flags.
4) If the overrun flag is active:
   a. Alert application that at least one error could not be managed.
   b. Clear the overrun flag.

The FLEXCAN internal processes can access memory in transactions larger than 32 bits. For the indication, this kind of access is considered a consecutive sequence of 32-bit accesses. If errors are found in two or more 32-bit words the interrupt and overrun flags are set simultaneously.

#### *60.6.16.3 Error Reporting*

The report registers Error Report Address register (RERRAR), Error Report Data register (RERRDR), and Error Report Syndrome register (RERRSYNR) provide detailed information about the address read, raw data, and syndrome read with error, and are indicated by the flags described in Error indication. The address, data, and syndrome registers are updated simultaneously along with the error flags, according to these rules:

1) If either of the two non-correctable error flags is currently set, the report registers are not updated (the previous non-correctable error reporting is preserved).
2) Otherwise (either no error flag is currently set or only the correctable error flag is currently set), the report registers are updated according to the new error; or according to the most severe of new errors if non-correctable and correctable errors are simultaneously detected.

Reporting of errors detected in accesses larger than 32-bit follows the rules described in Error indication and in the description of Error Report Address register (RERRAR).

The address reported in RERRAR and defined in ERRIAR are not the same listed in the module memory map. The relation between the reported addresses and the respective ones in the module memory map is shown in the description of the Error Injection Address register (ERRIAR).

Addresses reported when reading memory portions organized as FIFOs, such as the Legacy Rx FIFO Structure, Enhanced RX FIFO Structure and the Rx FIFO Information Register (RXFIR), refer to the address of the specific entry accessed in the FIFO, not to the FIFO base address.

To assure coherence of the error report registers it is necessary to turn off the report update by setting MECR[RERRDIS] before reading the report registers.

#### 60.6.16.4 Response to Errors

Correctable errors have no consequence on FLEXCAN operation because affected data is corrected before its use by the host or FLEXCAN internal processes.

For host-initiated reads, a non-correctable error may affect the host, but does not affect FLEXCAN operation.

Non-correctable errors detected on memory reads requested by the FLEXCAN internal processes may result in incorrect operation depending on the state of MECR[NCEFAFRZ], as follows:

- During reception (either matching or move-in processes), when a non-correctable error occurs, an incorrect destination may be selected to store the incoming frame, a corrupted frame may be stored in the correct destination or both. If MECR[NCEFAFRZ] is set, FLEXCAN stops operation automatically and enters in Freeze mode to prevent corrupted data from being treated as valid by FLEXCAN internal processes. When MECR[NCEFAFRZ] is negated, FLEXCAN continues working and a corrupted frame is received.
- During the arbitration process, when a non-correctable error occurs, either a non-highest priority Tx message buffer may be mistakenly selected for transmission or its data may be corrupted. If MECR[NCEFAFRZ] is set, FLEXCAN stops operation automatically and enters Freeze mode before starting the move-out. When MECR[NCEFAFRZ] is negated, FLEXCAN proceeds to move out with a corrupted frame that will be transmitted on the FLEXCAN bus.
- During the move-out process, when a non-correctable error occurs, a corrupted frame is copied from the selected Tx MB that won the arbitration to the Tx SMB for transmission. If MECR[NCEFAFRZ] is set, FLEXCAN stops operation automatically and enters Freeze mode before starting the transmission. When MECR[NCEFAFRZ] is negated, the corrupted frame is transferred from the Tx SMB to the Protocol Engine (PE) sub-block and is transmitted on the FLEXCAN bus.
- A non-correctable error can also be detected beyond the move-out process, when Tx data is read from Tx SMB (buffer located in RAM) to be transferred to the PE subblock for transmission. In this case, a frame with corrupted ID and/or data is transmitted on the FLEXCAN bus.

To prevent the frame from being successfully received by the external nodes, FLEXCAN inverts all bits in the CRC field (CRC sequence plus CRC delimiter), and transmits an error flag just after CRC delimiter as a result of self-detecting a Bit1 error and a form error due to the CRC field inversion. When MECR[NCEFAFRZ] is set, FLEXCAN stops operation automatically and enters Freeze mode just after the error frame. When MECR[NCEFAFRZ] is negated, FLEXCAN may attempt to re-transmit the same frame, as long as no other higher priority Tx MB is subsequently configured for transmission.

In the event that the non-correctable error persists, FLEXCAN eventually reaches the Bus Off state because of consecutive error detections. ECR[TXERRCNT] is updated every time FLEXCAN inverts the CRC field, causing errors as described above.

When MECR[NCEFAFRZ] is set and FLEXCAN enters Freeze mode, only the CPU can cause FLEXCAN to exit Freeze mode and resume Normal mode. The assertion of MECR[NCEFAFRZ] is the only way to prevent corrupted frames from being transmitted on the FLEXCAN bus up to the move-out internal process.

#### 60.6.16.5 Error Injection

The error injection registers ERRIAR, ERRIDPR, and ERRIPPR are used to inject errors in memory reads in order to force errors and consequently update the indication and reporting registers. The relation between the error injection addresses and the respective ones in the module memory map is shown in the description of the Error Injection Address register (ERRIAR).

The injection is done by flipping the data and parity bits corresponding to the bits in 1 in ERRIDPR and ERRIPPR. The injection can be selected specifically for memory accesses requested by the host or by FLEXCAN internal processes.

In case of accesses larger than 32-bits, the EXTERRIE bit in the Memory Error Control register (MECR) extends the injection pattern, replicating it in 32-bit words to fill the width of the access.

The error report registers can provide information to the application for customized management of these situations.

***Note:*** It is very unlikely, but error injection may correct a bit with error. This will not raise the error flags and reports as expected.

To ensure coherence among error injection registers and avoid spurious error injections, MECR[HAERRIE] and MECR[FAERRIE] must be cleared when configuring the memory injection registers.

### 60.6.17 Clock Domains and Restrictions

The FLEXCAN module has two clock domains asynchronous to each other:

- The bus domain feeds the Control Host Interface (CHI) submodule and is derived from the peripheral clock.
- The oscillator domain feeds the CAN Protocol Engine (PE) submodule and is derived directly from a crystal oscillator clock, so that very low jitter performance can be achieved on the FLEXCAN bus.

When CTRL1[CLKSRC] is set, synchronous operation occurs because both domains are connected to the peripheral clock (creating a 1:1 ratio between the peripheral and oscillator clocks).

When the two domains are connected to clocks with different frequencies and/or phases, there are restrictions on the frequency relationship between the two clock domains. In the case of asynchronous operation, the bus domain clock frequency must always be greater than the oscillator domain clock frequency.

***Note:*** Asynchronous operation with a 1:1 ratio between peripheral and oscillator clocks is not allowed.

## 60.7 Register and RAM Special

### 60.7.1 Registers in RAM

The table below lists the registers in RAM.

**Table 103. FLEXCAN Registers in RAM**

| Register | Note |
|---|---|
| Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK) | If the CPU is not initialized, the reset value of these registers is 0xFFFFFFFF. |
| Rx 14 Mask Register (FLEXCAN_RX14MASK) | |
| Rx 15 Mask Register (FLEXCAN_RX15MASK) | |
| Rx FIFO Global Mask Register (FLEXCAN_RXFGMASK) | |
| Message buffers (FLEXCAN_RAMn) | N/A |
| Rx Individual Mask Registers (FLEXCAN_RXIMRn) | N/A |
| Enhanced Rx FIFO (FLEXCAN_ERX_FIFOn) | N/A |
| Enhanced Rx FIFO Filter Element (FLEXCAN_ERFFEL) | N/A |

### 60.7.2 Register Access and Reset

The table below lists the register access and reset information.

**Table 104. Register access and reset information**

| Register | Access Type* | Affected by Hard Reset | Affected by Soft Reset |
|---|---|---|---|
| *Module Configuration Register (FLEXCAN_MCR)* | S | Yes | Yes |
| *Control 1 Register (FLEXCAN_CTRL1)* | S/U | Yes | No |
| *Free Running Timer Register (FLEXCAN_TIMER)* | S/U | Yes | Yes |
| *Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK)* | S/U | No | No |
| *Rx 14 Mask Register (FLEXCAN_RX14MASK)* | S/U | No | No |
| *Rx 15 Mask Register (FLEXCAN_RX15MASK)* | S/U | No | No |
| *Error Counter Register (FLEXCAN_ECR)* | S/U | Yes | Yes |
| *Error and Status 1 Register (FLEXCAN_ESR1)* | S/U | Yes | Yes |
| *Interrupt Masks 1 Register (FLEXCAN_IMASK1)* | S/U | Yes | Yes |
| *Interrupt Flags 1 Register (FLEXCAN_IFLAG1)* | S/U | Yes | Yes |
| *Control 2 Register (FLEXCAN_CTRL2)* | S/U | Yes | No |
| *Error and Status 2 Register (FLEXCAN_ESR2)* | S/U | Yes | Yes |
| *CRC Register (FLEXCAN_CRCR)* | S/U | Yes | Yes |
| *Rx FIFO Global Mask Register (FLEXCAN_RXFGMASK)* | S/U | No | No |
| *Rx FIFO Information Register (FLEXCAN_RXFIR)* | S/U | No | No |
| *CAN Bit Timing Register (FLEXCAN_CBT)* | S/U | Yes | No |
| *Message buffers (FLEXCAN_RAMn)* | S/U | No | No |
| *Rx Individual Mask Registers (FLEXCAN_RXIMRn)* | S/U | No | No |
| *Memory Error Control Register (FLEXCAN_MECR)* | S/U | Yes | Yes |
| *Error Injection Address Register (FLEXCAN_ERRIAR)* | S/U | Yes | Yes |
| *Error Injection Data Pattern Register (FLEXCAN_ERRIDPR)* | S/U | Yes | Yes |
| *Error Injection Parity Pattern Register (FLEXCAN_ERRIPPR)* | S/U | Yes | Yes |
| *Error Report Address Register (FLEXCAN_RERRAR)* | S/U | Yes | Yes |
| *Error Report Data Register (FLEXCAN_RERRDR)* | S/U | Yes | Yes |
| *Error Report Syndrome Register (FLEXCAN_RERRSYNR)* | S/U | Yes | Yes |
| *Error Status Register (FLEXCAN_ERRSR)* | S/U | Yes | Yes |
| *Pretended Networking Control 1 Register (FLEXCAN_CTRL1_PN)* | S/U | Yes | Yes |
| *Pretended Networking Control 2 Register (FLEXCAN_CTRL2_PN)* | S/U | Yes | Yes |
| *Pretended Networking Wake Up Match Register (FLEXCAN_WU_MTC)* | S/U | Yes | Yes |

| | | | |
|---|---|---|---|
| *Pretended Networking ID Filter 1 Register (FLEXCAN_FLT_ID1)* | S/U | Yes | Yes |
| *Pretended Networking DLC Filter Register (FLEXCAN_FLT_DLC)* | S/U | Yes | Yes |
| *Pretended Networking Payload Low Filter 1 Register (FLEXCAN_PL1_LO)* | S/U | Yes | Yes |
| *Pretended Networking Payload High Filter 1 Register (FLEXCAN_PL1_HI)* | S/U | Yes | Yes |
| *Pretended Networking ID Filter 2 Register/ ID Mask Register (FLEXCAN_FLT_ID2_IDMASK)* | S/U | Yes | Yes |
| *Pretended Networking Payload Low Filter 2 Register / Payload Low Mask Register (FLEXCAN_PL2_PLMASK_LO)* | S/U | Yes | Yes |
| *Pretended Networking Payload High Filter 2 High Order Bits / Payload High Mask Register (FLEXCAN_PL2_PLMASK_HI)* | S/U | Yes | Yes |
| *Pretended Networking Wake Up Message Buffer 0 registers (FLEXCAN_WMB0_ *)* | S/U | Yes | No |
| *Pretended Networking Wake Up Message Buffer 1 registers (FLEXCAN_WMB1_ *)* | S/U | Yes | No |
| *Pretended Networking Wake Up Message Buffer 2 registers (FLEXCAN_WMB2_ *)* | S/U | Yes | No |
| *Pretended Networking Wake Up Message Buffer 3 registers (FLEXCAN_WMB3_ *)* | S/U | Yes | No |
| *CAN FD Control Register (FLEXCAN_FDCTRL)* | S/U | Yes | No |
| *CAN FD Bit Timing Register (FLEXCAN_FDCBT)* | S/U | Yes | No |
| *CAN FD CRC Register (FLEXCAN_FDCRC)* | S/U | Yes | Yes |
| *Enhanced Rx FIFO Control Register (FLEXCAN_ERFCR)* | S/U | Yes | Yes |
| *Enhanced Rx FIFO Interrupt Enable Register (FLEXCAN_ERFIER)* | S/U | Yes | Yes |
| *Enhanced Rx FIFO Status Register (FLEXCAN_ERFSR)* | S/U | Yes | Yes |
| *Enhanced Rx FIFO (FLEXCAN_ERX_FIFOn)* | S/U | No | No |
| *Enhanced Rx FIFO Filter Element (FLEXCAN_ERFFEL)* | S/U | No | No |

* S: Supervisor mode; U: User mode.

### 60.7.3 Registers Supported in Chip

**Table 105. Registers supported in chip**

| *Register* | Supported by Instances | Comments |
|---|---|---|
| *Module Configuration Register (FLEXCAN_MCR)* | FLEXCAN0-3 | |
| *Control 1 Register (FLEXCAN_CTRL1)* | FLEXCAN0-3 | |
| *Free Running Timer Register (FLEXCAN_TIMER)* | FLEXCAN0-3 | |
| *Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK)* | FLEXCAN0-3 | |
| *Rx 14 Mask Register (FLEXCAN_RX14MASK)* | FLEXCAN0-3 | |
| *Rx 15 Mask Register (FLEXCAN_RX15MASK)* | FLEXCAN0-3 | |

| | | |
|---|---|---|
| *Error Counter Register (FLEXCAN_ECR)* | FLEXCAN0-3 | |
| *Error and Status 1 Register (FLEXCAN_ESR1)* | FLEXCAN0-3 | |
| *Interrupt Masks 1 Register (FLEXCAN_IMASK1)* | FLEXCAN0-3 | |
| *Interrupt Flags 1 Register (FLEXCAN_IFLAG1)* | FLEXCAN0-3 | |
| *Control 2 Register (FLEXCAN_CTRL2)* | FLEXCAN0-3 | |
| *Error and Status 2 Register (FLEXCAN_ESR2)* | FLEXCAN0-3 | |
| *CRC Register (FLEXCAN_CRCR)* | FLEXCAN0-3 | |
| *Rx FIFO Global Mask Register (FLEXCAN_RXFGMASK)* | FLEXCAN0-3 | |
| *Rx FIFO Information Register (FLEXCAN_RXFIR)* | FLEXCAN0-3 | |
| *CAN Bit Timing Register (FLEXCAN_CBT)* | FLEXCAN0-3 | |
| *Message buffers (FLEXCAN_RAMn)* | FLEXCAN0-3 | |
| *Rx Individual Mask Registers (FLEXCAN_RXIMRn)* | FLEXCAN0-3 | |
| *Memory Error Control Register (FLEXCAN_MECR)* | FLEXCAN0-3 | |
| *Error Injection Address Register (FLEXCAN_ERRIAR)* | FLEXCAN0-3 | |
| *Error Injection Data Pattern Register (FLEXCAN_ERRIDPR)* | FLEXCAN0-3 | |
| *Error Injection Parity Pattern Register (FLEXCAN_ERRIPPR)* | FLEXCAN0-3 | |
| *Error Report Address Register (FLEXCAN_RERRAR)* | FLEXCAN0-3 | |
| *Error Report Data Register (FLEXCAN_RERRDR)* | FLEXCAN0-3 | |
| *Error Report Syndrome Register (FLEXCAN_RERRSYNR)* | FLEXCAN0-3 | |
| *Error Status Register (FLEXCAN_ERRSR)* | FLEXCAN0-3 | |
| *Pretended Networking Control 1 Register (FLEXCAN_CTRL1_PN)* | FLEXCAN0/1/2/3 | |
| *Pretended Networking Control 2 Register (FLEXCAN_CTRL2_PN)* | FLEXCAN0/1/2/3 | |
| *Pretended Networking Wake Up Match Register (FLEXCAN_WU_MTC)* | FLEXCAN0/1/2/3 | |
| *Pretended Networking ID Filter 1 Register (FLEXCAN_FLT_ID1)* | FLEXCAN0/1/2/3 | |
| *Pretended Networking DLC Filter Register (FLEXCAN_FLT_DLC)* | FLEXCAN0/1/2/3 | |
| *Pretended Networking Payload Low Filter 1 Register (FLEXCAN_PL1_LO)* | FLEXCAN0/1/2/3 | |
| *Pretended Networking Payload High Filter 1 Register (FLEXCAN_PL1_HI)* | FLEXCAN0/1/2/3 | |
| *Pretended Networking ID Filter 2 Register/ ID Mask Register (FLEXCAN_FLT_ID2_IDMASK)* | FLEXCAN0/1/2/3 | |
| *Pretended Networking Payload Low Filter 2 Register / Payload Low Mask Register (FLEXCAN_PL2_PLMASK_LO)* | FLEXCAN0/1/2/3 | |

| | |
|---|---|
| *Pretended Networking Payload High Filter 2 High Order Bits / Payload High Mask Register (FLEXCAN_PL2_PLMASK_HI)* | FLEXCAN0/1/2/3 |
| *Pretended Networking Wake Up Message Buffer 0 registers (FLEXCAN_WMB0_*)* | FLEXCAN0/1/2/3 |
| *Pretended Networking Wake Up Message Buffer 1 registers (FLEXCAN_WMB1_*)* | FLEXCAN0/1/2/3 |
| *Pretended Networking Wake Up Message Buffer 2 registers (FLEXCAN_WMB2_*)* | FLEXCAN0/1/2/3 |
| *Pretended Networking Wake Up Message Buffer 3 registers (FLEXCAN_WMB3_*)* | FLEXCAN0/1/2/3 |
| *CAN FD Control Register (FLEXCAN_FDCTRL)* | FLEXCAN0-3 |
| *CAN FD Bit Timing Register (FLEXCAN_FDCBT)* | FLEXCAN0-3 |
| *CAN FD CRC Register (FLEXCAN_FDCRC)* | FLEXCAN0-3 |
| *Enhanced Rx FIFO Control Register (ERFCR)* | FLEXCAN0-3 |
| *Enhanced Rx FIFO Interrupt Enable Register (ERFIER)* | FLEXCAN0-3 |
| *Enhanced Rx FIFO Status Register (ERFSR)* | FLEXCAN0-3 |
| *Enhanced Rx FIFO (FLEXCAN_ERX_FIFOn)* | FLEXCAN0-3 |
| *Enhanced Rx FIFO Filter Element (ERFFEL)* | FLEXCAN0-3 |

### 60.7.4 Message Buffer Definition

The following table shows the message buffer structure used by the FLEXCAN module. Both Extended (29-bit identifier) and Standard (11-bit identifier) frames used in the CAN specification (Version 2.0 Part B) are represented. Each MB is formed by 16, 24, 40, or 72 bytes, depending on the quantity of data bytes allocated for the message payload: 8, 16, 32, or 64 data bytes, respectively.

The memory area from 0x80 to 0x27F is used by the mailboxes. When FLEXCAN FD is enabled, the exact address for each MB depends on the size of its payload.

**Table 106. Message buffer structure example with 64 Bytes payload**

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0[1] | EDL | BRS | ESI | | CODE | | | | | SRR | IDE | RTR | DLC | | | | TIME STAMP | | | | | | | | | | | | | | | |
| 0x4[2] | PRIO[3] | | | | | | | | ID (standard/extended) | | | | | | | | ID (extended) | | | | | | | | | | | | | | | |
| 0x8 | Data Byte 0 | | | | | | | | Data Byte 1 | | | | | | | | Data Byte 2 | | | | | | | | Data Byte 3 | | | | | | | |
| 0xC | Data Byte 4 | | | | | | | | Data Byte 5 | | | | | | | | Data Byte 6 | | | | | | | | Data Byte 7 | | | | | | | |
| 0x10 | Data Byte 8 | | | | | | | | Data Byte 9 | | | | | | | | Data Byte 10 | | | | | | | | Data Byte 11 | | | | | | | |
| 0x14 | Data Byte 12 | | | | | | | | Data Byte 13 | | | | | | | | Data Byte 14 | | | | | | | | Data Byte 15 | | | | | | | |
| 0x18 | Data Byte 16 | | | | | | | | Data Byte 17 | | | | | | | | Data Byte 18 | | | | | | | | Data Byte 19 | | | | | | | |

| | | | | |
|---|---|---|---|---|
| 0x1C | Data Byte 20 | Data Byte 21 | Data Byte 22 | Data Byte 23 |
| 0x20 | Data Byte 24 | Data Byte 25 | Data Byte 26 | Data Byte 27 |
| 0x24 | Data Byte 28 | Data Byte 29 | Data Byte 30 | Data Byte 31 |
| 0x28 | Data Byte 32 | Data Byte 33 | Data Byte 34 | Data Byte 35 |
| 0x2C | Data Byte 36 | Data Byte 37 | Data Byte 38 | Data Byte 39 |
| 0x30 | Data Byte 40 | Data Byte 41 | Data Byte 42 | Data Byte 43 |
| 0x34 | Data Byte 44 | Data Byte 45 | Data Byte 46 | Data Byte 47 |
| 0x38 | Data Byte 48 | Data Byte 49 | Data Byte 50 | Data Byte 51 |
| 0x3C | Data Byte 52 | Data Byte 53 | Data Byte 54 | Data Byte 55 |
| 0x40 | Data Byte 56 | Data Byte 57 | Data Byte 58 | Data Byte 59 |
| 0x44 | Data Byte 60 | Data Byte 61 | Data Byte 62 | Data Byte 63 |

***Note:***

1. The 32 bits corresponding to 0x0 are defined as "C/S".
2. The 32 bits corresponding to 0x4 are defined as "ID".
3. Only valid for Tx mailboxes.

**EDL – Extended Data Length**

The EDL bit distinguishes between CAN format and CAN FD format frames. EDL must not be set for message buffers configured to RANSWER with code field 0b1010.

**BRS – Bit Rate Switch**

The BRS bit defines whether the bit rate is switched inside a CAN FD format frame.

**ESI – Error State Indicator**

The ESI bit indicates if the transmitting node is error active or error passive.

**CODE – Message Buffer Code**

The CODE field can be accessed (read or write) by the CPU and by the FLEXCAN module, as part of the message buffer matching and arbitration process. The encoding is shown in the following table.

**Table 107. Message buffer code for Rx buffers**

| Rx CODE | BEFORE Receiving New Frame | SRV(1) | AFTER Successful Reception(2) | RRS(3) | Comment |
|---|---|---|---|---|---|
| 0000: INACTIVE – MB is not active. | INACTIVE | — | — | — | MB does not participate in the matching process. |
| 0100: EMPTY – MB is active and empty. | EMPTY | — | FULL | — | When a frame is received successfully (after the Move-in process), the CODE field is automatically updated to FULL. |

| Rx CODE | BEFORE Receiving New Frame | SRV(1) | AFTER Successful Reception(2) | RRS(3) | Comment |
|---|---|---|---|---|---|
| 0010: FULL – MB is full. | FULL | Yes | FULL | — | The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a new frame is moved to the MB after the MB was serviced, the code remains FULL. |
| | | No | OVERRUN | — | If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. |
| 0110: OVERRUN – MB is being overwritten into a full buffer. | OVERRUN | Yes | FULL | — | If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB, the code returns to FULL. |
| | | No | OVERRUN | — | If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. |
| 1010: RANSWER(4) – A frame was configured to recognize a Remote Request Frame and transmit a Response Frame in return.(5) | RANSWER | — | TANSWER (1110) | 0 | A Remote Answer was configured to recognize a remote request frame received. After that, an MB is set to transmit a response frame. The code is automatically changed to TANSWER (1110). |
| | | — | — | 1 | This code is ignored during the matching and arbitration process. |
| CODE [0]=1: BUSY – FLEXCAN is updating the contents of the MB. The CPU must not access the MB. | BUSY(6) | — | FULL | — | This code is ignored during the matching and arbitration process. |

1) SRV: Serviced MB. MB was read and unlocked by reading FLEXCAN_TIMER or other MB.
2) A frame is considered a successful reception after the frame is moved to an MB.
3) Remote Request Stored bit.
4) Code 1010 is not considered Tx and an MB with this code should not be aborted.
5) Code 1010 must be used in message buffers configured in CAN FD format, having the EDL bit set.
6) For Tx MBs, the BUSY bit should be ignored upon read, except when FLEXCAN_MCR.AEN is set. If this bitis asserted, the corresponding MB does not participate in the matching process.

**Table 108. Message buffer code for Tx buffers**

| Tx CODE Description | BEFORE Tx Frame | MB RTR | AFTER Successful Transmission | Comment |
|---|---|---|---|---|
| 1000: INACTIVE –MB is not active. | INACTIVE | — | — | MB does not participate in the arbitration process. |
| 1001: ABORT – MB is aborted. | ABORT | — | — | MB does not participate in the arbitration process. |
| 1100: DATA – MB is a Tx Data Frame (MB RTR must be 0). | DATA | 0 | INACTIVE | Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state. |
| 1100: REMOTE – MB is a Tx Remote Request Frame (MB RTR must be 1). | REMOTE | 1 | EMPTY | Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID. |
| 1110: TANSWER –MB is a Tx Response Frame from an incoming Remote Request Frame. | TANSWER | — | RANSWER | This is an intermediate code that is automatically written to the MB by the CHI as a result of a match to a remote request frame. The remote response frame will be transmitted unconditionally once, and then the code will automatically return to RANSWER (1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. See "Matching Process" and "Arbitration Process" for details. |

**SRR — Substitute Remote Request**

Fixed recessive bit, used only in extended format. It must be set to one by the user for transmission (Tx Buffers) and will be stored with the value received on the FLEXCAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FLEXCAN receives this bit as dominant, then it is interpreted as an arbitration loss.

1 = Recessive value is compulsory for transmission in extended format frames

0 = Dominant is not a valid value for transmission in extended format frames

**IDE — ID Extended Bit**

This field identifies whether the frame format is standard or extended.

1 = Frame format is extended

0 = Frame format is standard

**RTR — Remote Transmission Request**

This bit affects the behavior of remote frames and is part of the reception filter. See the description of the RRS field in the Control 2 register (CTRL2) for additional details.

If FLEXCAN transmits this bit as "1" (recessive) and receives it as "0" (dominant), it is interpreted as an arbitration loss. If this bit is transmitted as "0" (dominant), then if it is received as "1" (recessive), the FLEXCAN module treats it as a bit error. If the value received matches the value transmitted, it is considered a successful bit transmission.

1 = Indicates the current MB may have a remote request frame to be transmitted if MB is Tx. If the MB is Rx then incoming remote request frames may be stored.

0 = Indicates the current MB has a data frame to be transmitted. In Rx MB it may be considered in matching processes.

***Note:*** When configuring CAN FD frames, the RTR bit must be negated.

**DLC — Length of Data in Bytes**

This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space (see Table 55-10). In reception, this field is written by the FLEXCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field (see Table 55-13).

**TIME STAMP — Free-Running Counter Time Stamp**

This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the FLEXCAN bus.

**PRIO — Local priority**

This 3-bit field is used only when MCR[LPRIO_EN] is set, and it only makes sense for Tx mailboxes. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Arbitration process.

**ID — Frame Identifier**

In standard frame format, only the 11 most significant bits (28 to 18) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification in both receive and transmit cases.

**DATA BYTE 0 to 63 — Data Field**

Up to sixty-four bytes can be used for a data frame, depending on the size of the payload selected for the message buffers.

For Rx frames, the data is stored as it is received from the FLEXCAN bus. DATA BYTE (n) is valid only if n is less than DLC as shown in the table below.

**Table 109. Data Bytes Validity**

| DLC | Valid Data Bytes |
|---|---|
| 0 | none |
| 1 | DATA BYTE 0 |
| 2 | DATA BYTE 0 to 1 |
| 3 | DATA BYTE 0 to 2 |
| 4 | DATA BYTE 0 to 3 |
| 5 | DATA BYTE 0 to 4 |
| 6 | DATA BYTE 0 to 5 |
| 7 | DATA BYTE 0 to 6 |
| 8 | DATA BYTE 0 to 7 |

| | |
|---|---|
| 9 | DATA BYTE 0 to 11 |
| 10 | DATA BYTE 0 to 15 |
| 11 | DATA BYTE 0 to 19 |
| 12 | DATA BYTE 0 to 23 |
| 13 | DATA BYTE 0 to 31 |
| 14 | DATA BYTE 0 to 47 |
| 15 | DATA BYTE 0 to 63 |

### 60.7.5 Message Buffer Memory Map

8-byte message buffer is as follows.

**Table 110. 8-byte message buffer**

| Address | MB ID |
|---|---|
| 0x80 | MB0 |
| 0x90 | MB1 |
| 0xA0 | MB2 |
| 0xB0 | MB3 |
| 0xC0 | MB4 |
| 0xD0 | MB5 |
| 0xE0 | MB6 |
| 0xF0 | MB7 |
| 0x100 | MB8 |
| 0x110 | MB9 |
| 0x120 | MB10 |
| 0x130 | MB11 |
| 0x140 | MB12 |
| 0x150 | MB13 |
| 0x160 | MB14 |
| 0x170 | MB15 |
| 0x180 | MB16 |
| 0x190 | MB17 |
| 0x1A0 | MB18 |
| 0x1B0 | MB19 |
| 0x1C0 | MB20 |
| 0x1D0 | MB21 |
| 0x1E0 | MB22 |
| 0x1F0 | MB23 |
| 0x200 | MB24 |
| 0x210 | MB25 |
| 0x220 | MB26 |
| 0x230 | MB27 |
| 0x240 | MB28 |
| 0x250 | MB29 |

Flagchip

| | |
|---|---|
| 0x260 | MB30 |
| 0x270 | MB31 |

### 60.7.6 Legacy Rx FIFO Structure

When the FLEXCAN_MCR[RFEN] bit is set, the memory area from 0x80 to 0xDC (which is normally occupied by MBs 0–5) is used by the Rx FIFO engine.

The region 0x80–0x8C contains the output of the FIFO, which must be read by the CPU as a message buffer. The FIFO output contains the oldest message that has been received but not yet read. The region 0x90–0xDC is reserved for internal use of the FIFO engine.

An additional memory area, which starts at 0xE0 and may extend up to 0x27C (normally occupied by MBs 6–31) depending on the FLEXCAN_CTRL2[RFFN] field setting, contains the ID filter table (configurable from 8 to 128 table elements), which specifies filtering criteria for accepting frames into the FIFO.

Out of reset, the ID filter table flexible memory area defaults to 0xE0 and extends only to 0xFC, which corresponds to MBs 6 to 7 for RFFN = 0, for backward compatibility with previous versions of the FLEXCAN.

**Table 111. Legacy Rx FIFO data structure**

| | 31 30 29 28 27 26 25 24 | 23 22 21 20 19 18 17 16 | 15 14 13 12 11 10 9 8 | 7 6 5 4 3 2 1 0 |
|---|---|---|---|---|
| 0x80 | IDHIT | SRR / IDE / RTR / DLC | TIME STAMP | |
| 0x84 | ID (standard/extended) | | ID (extended) | |
| 0x88 | Data Byte 0 | Data Byte 1 | Data Byte 2 | Data Byte 3 |
| 0x8C | Data Byte 4 | Data Byte 5 | Data Byte 6 | Data Byte 7 |
| 0x90 To 0xDC | Reserved for CPU | | | |
| 0xE0 | ID filter table element 0 | | | |
| 0xE4 To 0x2D8 | ID filter table element 1 to 126 | | | |
| 0x2DC | ID filter table element 127 | | | |

Each ID filter table element occupies an entire 32-bit word and can be compounded by one, two, or four Identifier Acceptance Filters (IDAF) depending on the MCR[IDAM] field setting. The following figures show the IDAF indexation.

The following table shows the three different formats of the ID table elements. Note that all elements of the table must have the same format.

**Table 112. ID filter table structure**

| | 31 30 29 28 27 26 25 24 | 23 22 21 20 19 18 17 16 | 15 14 13 12 11 10 9 8 | 7 6 5 4 3 2 1 0 |
|---|---|---|---|---|
| A | RTR / IDE | RXIDA (standard: bit29 - bit19, extended: bit29 – bit1) | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| B | RTR | IDE | RXIDB_0 (standard: bit29 - bit19, extended: bit29 – bit16) | | RTR | IDE | RXIDB_1 (standard: bit13 – bit3, extended: bit13 – bit0) |
| C | RXIDC_0 (std/ext: bit31 – bit24) | | RXIDC_1 (std/ext: bit23– bit16) | | RXIDC_2 (std/ext: bit15 – bit8) | | RXIDC_3 (std/ext: bit7 – bit0) |

**RTR — Remote frame**

This bit specifies if remote frames are accepted into the Legacy FIFO if they match the target ID.

1 = Remote frames can be accepted, and data frames are rejected

0 = Remote frames are rejected, and data frames can be accepted

**IDE — Extended frame**

Specifies whether extended or standard frames are accepted into the Legacy FIFO if they match the target ID.

1 = Extended frames can be accepted, and standard frames are rejected

0 = Extended frames are rejected, and standard frames can be accepted

**RXIDA — Rx frame identifier (Format A)**

Specifies an ID to be used as acceptance criteria for the Legacy FIFO. In the standard frame format, only the 11 most significant bits (29 to 19) are used for frame identification. In the extended frame format, all bits are used.

**RXIDB_0, RXIDB_1 — Rx frame identifier (Format B)**

Specifies an ID to be used as acceptance criteria for the Legacy FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (29 to 19 and 13 to 3) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.

**RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 — Rx frame identifier (Format C)**

Specifies an ID to be used as acceptance criteria for the Legacy FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

**IDHIT — Identifier acceptance filter hit indicator**

This 9-bit field indicates which identifier acceptance filter was hit by the received message that is in the output of the Legacy Rx FIFO.

### 60.7.7 Enhanced Rx (ERX) FIFO Structure

When ERFCR[ERFEN] is set, the Enhanced Rx FIFO is enabled. The region 2000h–204Ch contains the output of the Enhanced Rx FIFO which must be read by the CPU as a message buffer. This output contains the oldest message that has been received but not yet read. The following table shows the Enhanced Rx FIFO data structure.

**Table 113. Enhanced Rx FIFO data structure**

| | 31 30 29 | 28 27 26 25 24 | 23 22 21 | 20 19 18 17 16 | 15 14 13 12 11 10 9 8 | 7 6 5 4 3 2 1 0 |
|---|---|---|---|---|---|---|
| 0x2000 | EDL, BRS, ESI | | SRR, IDE, RTR | DLC | TIME STAMP | |
| 0x2004 | | ID (standard/extended) | | | ID (extended) | |
| 0x2008 | Data Byte 0 | | Data Byte 1 | | Data Byte 2 | Data Byte 3 |
| 0x200C | Data Byte 4 | | Data Byte 5 | | Data Byte 6 | Data Byte 7 |

| | | | | |
|---|---|---|---|---|
| 0x2010 | Data Byte 8 | Data Byte 9 | Data Byte 10 | Data Byte 11 |
| 0x2014 | Data Byte 12 | Data Byte 13 | Data Byte 14 | Data Byte 15 |
| 0x2018 | Data Byte 16 | Data Byte 17 | Data Byte 18 | Data Byte 19 |
| 0x201C | Data Byte 20 | Data Byte 21 | Data Byte 22 | Data Byte 23 |
| 0x2020 | Data Byte 24 | Data Byte 25 | Data Byte 26 | Data Byte 27 |
| 0x2024 | Data Byte 28 | Data Byte 29 | Data Byte 30 | Data Byte 31 |
| 0x2028 | Data Byte 32 | Data Byte 33 | Data Byte 34 | Data Byte 35 |
| 0x202C | Data Byte 36 | Data Byte 37 | Data Byte 38 | Data Byte 39 |
| 0x2030 | Data Byte 40 | Data Byte 41 | Data Byte 42 | Data Byte 43 |
| 0x2034 | Data Byte 44 | Data Byte 45 | Data Byte 46 | Data Byte 47 |
| 0x2038 | Data Byte 48 | Data Byte 49 | Data Byte 50 | Data Byte 51 |
| 0x203C | Data Byte 52 | Data Byte 53 | Data Byte 54 | Data Byte 55 |
| 0x2040 | Data Byte 56 | Data Byte 57 | Data Byte 58 | Data Byte 59 |
| 0x2044 | Data Byte 60 | Data Byte 61 | Data Byte 62 | Data Byte 63 |
| 0x2048 | | | | IDHIT [8:0] |
| 0x204C | Reserved | | | |
| 0X2050 To 0x23BC | 11 Enhanced FIFO Elements (Reserved for CPU) | | | |

The table below lists ID HIT offset and high-resolution time stamp offset.

**Table 114. ID HIT and HR time stamp offsets**

| DLC | ID HIT offset | Time Stamp offset |
|---|---|---|
| 0 | 0x2008 | 0x200C |
| 1-4 | 0x200C | 0x2010 |
| 5-8 | 0x2010 | 0x2014 |
| 9 | 0x2014 | 0x2018 |
| 10 | 0x2018 | 0x201C |
| 11 | 0x201C | 0x2020 |
| 12 | 0x2020 | 0x2024 |
| 13 | 0x2028 | 0x202C |
| 14 | 0x2038 | 0x203C |
| 15 | 0x2048 | 0x204C |

### 60.7.8 Enhanced Rx FIFO ID Match Example

The ERFCR[NFE] control the total valid number of filter elements, FC7240 series supports total 32 filter elements allocated in 0x3000-0x307C, so the max value of NFE is (32/2 – 1) = 15, the max ERFCR[NXFIE] is (15 + 1) = 16.

The following table shows a simple example of (NFE = 6, NXFIE = 5), the total valued id filter elements number is 14((NFE + 1) * 2), the extends id filter is 5(NXFIE), the standard id filter is 4 (14 – 5*2). Allocate and filter id as follows. For Enhanced Rx FIFO ID filter usage, refer to the Enhanced Rx FIFO Matching Process.

**Table 115. ID filter element example**

| Address | ID FILTER elements | ID filter |
|---|---|---|
| 0x3000 | ID FILTER element0 | Extend id filter 1[0] |
| 0x3004 | ID FILTER element1 | Extend id filter 1[1] |
| 0x3008 | ID FILTER element2 | Extend id filter 2[0] |
| 0x300C | ID FILTER element3 | Extend id filter 2[1] |
| 0x3010 | ID FILTER element4 | Extend id filter 3[0] |
| 0x3014 | ID FILTER element5 | Extend id filter 3[1] |
| 0x3018 | ID FILTER element6 | Extend id filter 4[0] |
| 0x301C | ID FILTER element7 | Extend id filter 4[1] |
| 0x3020 | ID FILTER element8 | Extend id filter 5[0] |
| 0x3024 | ID FILTER element9 | Extend id filter 5[1] |
| 0x3028 | ID FILTER element10 | standard id filter 1 |
| 0x302C | ID FILTER element11 | standard id filter 2 |
| 0x3030 | ID FILTER element12 | standard id filter 3 |
| 0x3034 | ID FILTER element13 | standard id filter 4 |
| 0x3038 | ID FILTER element14 | not used |
| 0x303C | ID FILTER element15 | not used |
| 0x3040 | ID FILTER element16 | not used |
| 0x3044 | ID FILTER element17 | not used |
| 0x3048 | ID FILTER element18 | not used |
| 0x304C | ID FILTER element19 | not used |
| 0x3050 | ID FILTER element20 | not used |
| 0x3054 | ID FILTER element21 | not used |
| 0x3058 | ID FILTER element22 | not used |
| 0x305C | ID FILTER element23 | not used |
| 0x3060 | ID FILTER element24 | not used |
| 0x3064 | ID FILTER element25 | not used |
| 0x3068 | ID FILTER element26 | not used |
| 0x306C | ID FILTER element27 | not used |
| 0x3070 | ID FILTER element28 | not used |
| 0x3074 | ID FILTER element29 | not used |
| 0x3078 | ID FILTER element30 | not used |
| 0x307C | ID FILTER element31 | not used |

Flagchip

## 60.8 Register Definition

FLEXCAN0 base address : 4008_0000h

FLEXCAN1 base address : 4008_4000h

FLEXCAN2 base address : 4048_0000h

FLEXCAN3 base address : 4048_4000h

**Register Overview - FLEXCAN**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | FLEXCAN_MCR (Module Configuration Register) | 32 | RW | D890_000Fh | page |
| **4h** | FLEXCAN_CTRL1 (Control 1 Register) | 32 | RW | 0000_0000h | page |
| **8h** | FLEXCAN_TIMER (Free Running Timer) | 32 | RW | 0000_0000h | page |
| **10h** | FLEXCAN_RXMGMASK (Rx Mailboxes Global Mask Register) | 32 | RW | 0000_0000h | page |
| **14h** | FLEXCAN_RX14MASK (Rx 14 Mask Register) | 32 | RW | 0000_0000h | page |
| **18h** | FLEXCAN_RX15MASK (Rx 15 Mask Register) | 32 | RW | 0000_0000h | page |
| **1Ch** | FLEXCAN_ECR (Error Counter Register) | 32 | RO | 0000_0000h | page |
| **20h** | FLEXCAN_ESR1 (Error and Status 1 Register) | 32 | W1C | 0000_0000h | page |
| **28h** | FLEXCAN_IMASK1 (Interrupt Masks 1 Register) | 32 | RW | 0000_0000h | page |
| **30h** | FLEXCAN_IFLAG1 (Interrupt Flags 1 Register) | 32 | W1C | 0000_0000h | page |
| **34h** | FLEXCAN_CTRL2 (Control 2 Register) | 32 | RW | 00A0_0000h | page |
| **38h** | FLEXCAN_ESR2 (Error and Status 2 Register) | 32 | RO | 0000_0000h | page |
| **44h** | FLEXCAN_CRCR (CRC Register) | 32 | RO | 0000_0000h | page |
| **48h** | FLEXCAN_RXFGMASK (Legacy Rx FIFO Global Mask Register) | 32 | RW | 0000_0000h | page |
| **4Ch** | FLEXCAN_RXFIR (Legacy Rx FIFO Information Register) | 32 | RO | 0000_0000h | page |
| **50h** | FLEXCAN_CBT (CAN Bit Timing Register) | 32 | RW | 0000_0000h | page |
| **80h+n*4h** | FLEXCAN_RAMn (Message Buffer Registers) | 32 | RW | 0000_0000h | page |
| **880h+n*4h** | FLEXCAN_RXIMRn (Rx Individual Mask registers) | 32 | RW | 0000_0000h | page |
| **AE0h** | FLEXCAN_MECR (Memory Error Control Register) | 32 | RW | 800C_0080h | page |
| **AE4h** | FLEXCAN_ERRIAR (Error Injection Address Register) | 32 | RW | 0000_0000h | page |
| **AE8h** | FLEXCAN_ERRIDPR (Error Injection Data Pattern Register) | 32 | RW | 0000_0000h | page |
| **AECh** | FLEXCAN_ERRIPPR (Error Injection Parity Pattern Register) | 32 | RW | 0000_0000h | page |
| **AF0h** | FLEXCAN_RERRAR (Error Report Address Register) | 32 | RO | 0000_0000h | page |
| **AF4h** | FLEXCAN_RERRDR (Error Report Data Register) | 32 | RO | 0000_0000h | page |

| | | | | | |
|---|---|---|---|---|---|
| **AF8h** | FLEXCAN_RERRSYNR (Error Report Syndrome Register) | 32 | RO | 0000_0000h | page |
| **AFCh** | FLEXCAN_ERRSR (Error Status Register) | 32 | W1C | 0000_0000h | page |
| **B00h** | FLEXCAN_CTRL1_PN (Pretended Networking Control 1 Register) | 32 | RW | 0000_0100h | page |
| **B04h** | FLEXCAN_CTRL2_PN (Pretended Networking Control 2 Register) | 32 | RW | 0000_0000h | page |
| **B08h** | FLEXCAN_WU_MTC (Pretended Networking Wake Up Match Register) | 32 | W1C | 0000_0000h | page |
| **B0Ch** | FLEXCAN_FLT_ID1 (Pretended Networking ID Filter 1 Register) | 32 | RW | 0000_0000h | page |
| **B10h** | FLEXCAN_FLT_DLC (Pretended Networking DLC Filter Register) | 32 | RW | 0000_0008h | page |
| **B14h** | FLEXCAN_PL1_LO (Pretended Networking Payload Low Filter 1 Register) | 32 | RW | 0000_0000h | page |
| **B18h** | FLEXCAN_PL1_HI (Pretended Networking Payload High Filter 1 Register) | 32 | RW | 0000_0000h | page |
| **B1Ch** | FLEXCAN_FLT_ID2_IDMASK (Pretended Networking ID Filter 2 Register/ ID Mask Register) | 32 | RW | 0000_0000h | page |
| **B20h** | FLEXCAN_PL2_PLMASK_LO (Pretended Networking Payload Low Filter 2 Register/ Payload Low Mask Register) | 32 | RW | 0000_0000h | page |
| **B24h** | FLEXCAN_PL2_PLMASK_HI (Pretended Networking Payload High Filter 2 Low Order Bits/ Payload High Mask Register) | 32 | RW | 0000_0000h | page |
| **B40h+n*10h** | FLEXCAN_WMB_CSn (Wake Up Message Buffer Register for C/S) | 32 | RO | 0000_0000h | page |
| **B44h+n*10h** | FLEXCAN_WMB_IDn (Wake Up Message Buffer Register for ID) | 32 | RO | 0000_0000h | page |
| **B48h+n*10h** | FLEXCAN_WMB_D03_n (Wake Up Message Buffer Register for Data 0-3) | 32 | RO | 0000_0000h | page |
| **B4Ch+n*10h** | FLEXCAN_WMB_D47_n (Wake Up Message Buffer Register Data 4-7) | 32 | RO | 0000_0000h | page |
| **BF0h** | FLEXCAN_EPRS (Enhanced CAN Bit Timing Prescalers Register) | 32 | RW | 0000_0000h | page |
| **BF4h** | FLEXCAN_ENCBT (Enhanced Nominal CAN Bit Timing Register) | 32 | RW | 0000_0000h | page |
| **BF8h** | FLEXCAN_EDCBT (Enhanced Data Phase CAN bit Timing Register) | 32 | RW | 0000_0000h | page |
| **BFCh** | FLEXCAN_ETDC (Enhanced Transceiver Delay Compensation Register) | 32 | RW | 0000_0000h | page |
| **C00h** | FLEXCAN_FDCTRL (CAN FD Control Register) | 32 | RW | 8000_0100h | page |
| **C04h** | FLEXCAN_FDCBT (CAN FD Bit Timing Register) | 32 | RW | 0000_0000h | page |
| **C08h** | FLEXCAN_FDCRC (CAN FD CRC Register) | 32 | RO | 0000_0000h | page |
| **C0Ch** | FLEXCAN_ERFCR (Enhanced Rx FIFO Control Register) | 32 | RW | 0000_0000h | page |

| | | | | | |
|---|---|---|---|---|---|
| **C10h** | FLEXCAN_ERFIER (Enhanced Rx FIFO Interrupt Enable Register) | 32 | RW | 0000_0000h | page |
| **C14h** | FLEXCAN_ERFSR (Enhanced Rx FIFO Status Register) | 32 | RW | 0000_0000h | page |
| **1A00h** | FLEXCAN_GATE_CS (Gateway Control Register) | 32 | RW | 0000_0000h | page |
| **1A10h+n*4h** | FLEXCAN_GATE_BUFFERn (Gateway Buffer Register) | 32 | RW | 0000_0000h | page |
| **1A80h+n*4h** | FLEXCAN_GATE_DUMMYn (Gateway Router Dummy Register) | 32 | RW | 0000_0000h | page |
| **2000h+n*4h** | FLEXCAN_ERX_FIFOn (Enhanced Rx FIFO Register) | 32 | RW | 0000_0000h | page |
| **3000h+n*4h** | FLEXCAN_ERFFELn (Enhanced Rx FIFO Filter Element Register) | 32 | RW | 0000_0000h | page |

### 60.8.1 FLEXCAN_MCR (Module Configuration Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MDIS | FRZ | RFEN | HALT | NOTRDY | 0 | SOFTRST | FRZACK | SUPV | 0 | WRNEN | LPMACK | 0 | | SRXDIS | IRMQ |
| RST | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DMA | PNET_EN | LPRIOEN | AEN | FDEN | 0 | IDAM | | 0 | MAXMB | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**FLEXCAN_MCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**MDIS** | RW | Module Disable – Disables or enables the FLEXCAN module<br>When disabled, the FLEXCAN disables the clocks to the PE and CHI submodules. MDIS is not affected by soft reset.<br>0b - Enable the FLEXCAN module.<br>1b - Disable the FLEXCAN module. |

| | | |
|---|---|---|
| **30**<br>**FRZ** | RW | Freeze Enable<br>The FRZ bit specifies the FLEXCAN behavior when MCR[HALT] is set or when Debug mode is requested at chip level. When FRZ is asserted, FLEXCAN is enabled to enter Freeze mode. Negation of this bit field causes FLEXCAN to exit from Freeze mode. This bit is set by hardware when a non-correctable error is detected (MECR[NCEFAFRZ] is asserted).<br>0b - Not enabled to enter Freeze mode.<br>1b - Enabled to enter Freeze mode. |
| **29**<br>**RFEN** | RW | Legacy Rx FIFO Enable<br>This bit controls whether the Legacy Rx FIFO feature is enabled or not. When RFEN is set, MBs 0 to 5 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xDC) is used by the FIFO engine as well as additional MBs (up to 32, depending on CTRL2[RFFN] setting) which are used as Legacy Rx FIFO ID filter table elements. RFEN also impacts the definition of the minimum number of peripheral clocks per CAN bit. This bit can be written in Freeze mode only, because it is blocked by hardware in other modes.<br>NOTE: This bit cannot be set when CAN FD operation is enabled (see FDEN bit).<br>NOTE: RFEN bit must not be set if ERFCR[ERFEN] is set.<br>NOTE: The descriptions of the field settings vary by module instance.<br>0b - Disable Rx FIFO.<br>1b - Enable Rx FIFO. |
| **28**<br>**HALT** | RW | Halt FLEXCAN<br>Assertion of this bit puts the FLEXCAN module into Freeze mode. The CPU should clear it after initializing the message buffers and the Control registers CTRL1 and CTRL2. No reception or transmission is performed by FLEXCAN before this bit is cleared. Freeze mode cannot be entered when FLEXCAN is in a low power mode. The HALT bit is set by hardware when a non-correctable error is detected and MECR[NCEFAFRZ] is asserted.<br>0b - No Freeze mode request.<br>1b - Enters Freeze mode if the FRZ bit is asserted. |
| **27**<br>**NOTRDY** | RO | FLEXCAN Not Ready<br>This read-only bit indicates that FLEXCAN is either in Disable mode or Freeze mode. It is negated when FLEXCAN has exited these modes. This bit is not affected by soft reset.<br>0b - FLEXCAN module is either in Normal mode, Listen-Only mode, or Loop-Back mode.<br>1b - FLEXCAN module is either in Disable mode or Freeze mode. |

| | | |
|---|---|---|
| **25**<br>**SOFTRST** | RW | Soft Reset<br>When SOFTRST is asserted, FLEXCAN resets its internal state machines and some of the memory- mapped registers.<br>SOFTRST can be asserted directly by the CPU when it writes to the MCR register. Because soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFTRST bit remains asserted when reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed.<br>Soft reset cannot be applied when clocks are shut down in a low power mode. The module should be first removed from low power mode, and then soft reset can be applied. This bit is not affected by soft reset.<br>0b - No reset request.<br>1b - Resets the registers affected by soft reset. |
| **24**<br>**FRZACK** | RO | Freeze Mode Acknowledge<br>This read-only bit indicates that FLEXCAN is in Freeze mode and its prescaler is stopped. The Freeze mode request cannot be granted until current transmission or reception processes have finished.<br>Therefore the software can poll the FRZACK bit to know when FLEXCAN has actually entered Freeze mode. If Freeze mode request is negated, then this bit is negated after the FLEXCAN prescaler is running again. If Freeze mode is requested when FLEXCAN is in a low power mode, then the FRZACK bit will be set only when the low-power mode is exited. See Freeze mode. This bit is not affected by soft reset.<br>NOTE: FRZACK will be asserted within 178 CAN bits from the Freeze mode request by the CPU, and negated within 2 CAN bits after the Freeze mode request removal (see Protocol timing).<br>0b - FLEXCAN not in Freeze mode, prescaler running.<br>1b - FLEXCAN in Freeze mode, prescaler stopped. |
| **23**<br>**SUPV** | RW | Supervisor Mode-Configures the FLEXCAN to be either in Supervisor or User mode:<br>The registers affected by SUPV are marked as S/U in the Access Type column.<br>The SUPV reset value is 1, so the affected registers start with Supervisor access only. SUPV can be written only in Freeze mode because it is blocked by hardware in other modes.<br>0b - FLEXCAN is in User mode. Affected registers allow both Supervisor and Unrestricted accesses.<br>1b - FLEXCAN is in Supervisor mode.Affected registers allow only Supervisor access. Unrestricted access behaves as though the access was done to an unimplemented register location. |

| | | |
|---|---|---|
| **21**<br>**WRNEN** | RW | Warning Interrupt Enable<br>When asserted, this bit enables the generation of the TWRNINT and RWRNINT flags in the Error and Status Register 1 (ESR1). If WRNEN is negated, the TWRNINT and RWRNINT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. This bit can be written in Freeze mode only, because it is blocked by hardware in other modes.<br>0b - TWRNINT and RWRNINT bits are zero, independent of the values in the error counters.<br>1b - TWRNINT and RWRNINT bits are set when the respective error counter transitions from less than 96 to greater than or equal to 96. |
| **20**<br>**LPMACK** | RO | Low-Power Mode Acknowledge<br>This read-only bit indicates that FLEXCAN is in a low-power mode (Disable mode ). A low-power mode cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPMACK bit to know when FLEXCAN has actually entered low power mode. This bit is not affected by soft reset.<br>NOTE: LPMACK will be asserted within 180 CAN bits from the low-power mode request by the CPU, and negated within 2 CAN bits after the low-power mode request removal (see Protocol timing).<br>0b - FLEXCAN is not in a low-power mode.<br>1b - FLEXCAN is in a low-power mode. |
| **17**<br>**SRXDIS** | RW | Self Reception Disable<br>This bit defines whether FLEXCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br>0b - Self-reception enabled.<br>1b - Self-reception disabled. |
| **16**<br>**IRMQ** | RW | Individual Rx Masking And Queue Enable<br>This bit indicates whether Rx matching process will be based either on individual masking and queue or on masking scheme with RXMGMASK, RX14MASK, RX15MASK, and RXFGMASK. This bit can be written in Freeze mode only because it is blocked by hardware in other modes.<br>0b - Individual Rx masking and queue feature are disabled. For backward compatibility with legacy applications, the reading of C/S word locks the MB even if it is EMPTY.<br>1b - Individual Rx masking and queue feature are enabled. |

| | | |
|---|---|---|
| **15**<br>**DMA** | RW | DMA Enable<br>DMA controls whether the DMA feature is enabled or not. The DMA feature can only be used in Legacy Rx FIFO or Enhanced Rx FIFO, so consequently MCR[RFEN] or ERFCR[ERFEN] must be asserted.<br>When DMA and RFEN are set, IFLAG1[BUF5I] generates the DMA request and no RX FIFO interrupt is generated. This bit can be written in Freeze mode only as it is blocked by hardware in other modes.<br>NOTE: The descriptions of the field settings vary by module instance. |
| **14**<br>**PNET_EN** | RW | Pretended Networking Enable<br>This bit enables the Pretended Networking functionality. When in Stop mode, PE sub-block is kept operational, making it able to process Rx message filtering as defined by the Pretended Networking configuration registers. See Receive process under Pretended Networking mode. This bit can be written in Freeze mode only.<br>NOTE: This field is not supported in every instance.<br>0b - Pretended Networking mode is disabled.<br>1b - Pretended Networking mode is enabled. |
| **13**<br>**LPRIOEN** | RW | Local Priority Enable<br>This bit is provided for backwards compatibility with legacy applications. It controls whether the local priority feature is enabled or not. It is used to expand the ID used during the arbitration process. With this expanded ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br>0b - Local Priority disabled.<br>1b - Local Priority enabled. |
| **12**<br>**AEN** | RW | Abort Enable<br>When asserted, this bit enables the Tx abort mechanism. This mechanism guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the FLEXCAN bus without notification. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br>NOTE: When MCR[AEN] is asserted, only the abort mechanism (see Transmission abort mechanism) must be used for updating mailboxes configured for transmission.<br>CAUTION: Writing the Abort code into Rx mailboxes can cause unpredictable results when MCR[AEN] is asserted.<br>0b - Abort disabled.<br>1b - Abort enabled. |

| | | |
|---|---|---|
| **11**<br>**FDEN** | RW | CAN FD operation enable<br>This bit enables the FLEXCAN with FLEXible Data rate (CAN FD) operation. This bit can be written in Freeze mode only.<br>NOTE: FLEXCAN is able to transmit FD frame format according to ISO 11898-1.<br>NOTE: The Legacy Rx FIFO Enable (RFEN) bit cannot be set if FDEN is asserted.<br>0b - CAN FD is disabled. FLEXCAN is able to receive and transmit messages in CAN 2.0 format.<br>1b - CAN FD is enabled. FLEXCAN is able to receive and transmit messages in both CAN FD and CAN 2.0 formats. |
| **9-8**<br>**IDAM** | RW | ID Acceptance Mode<br>This 2-bit field identifies the format of the Legacy Rx FIFO ID filter table elements. Note that all elements of the table are configured at the same time by this field (they are all the same format). See Legacy Rx FIFO structure. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>00b - Format A: One full ID (standard and extended) per ID filter table element.<br>01b - Format B – Two full standard IDs or two partial 14-bit (standard and extended) IDs per ID Filter Table element.<br>10b - Format C – Four partial 8-bit standard IDs per ID Filter Table element.<br>11b - Format D – All frames rejected. |
| **6-0**<br>**MAXMB** | RW | Number Of The Last Message Buffer<br>This 7-bit field defines the number of the last message buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to a 16 MB configuration. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Number of the last MB = MAXMB<br>NOTE: MAXMB must be programmed with a value smaller than or equal to the number of available message buffers, as described in FLEXCAN memory partition for CAN FD.<br>Additionally, the definition of MAXMB value must take into account the region of MBs occupied by Legacy Rx FIFO and its ID filters table space defined by CTRL2[RFFN]. MAXMB also impacts the definition of the minimum number of peripheral clocks per CAN bit. |

### 60.8.2 FLEXCAN_CTRL1 (Control 1 Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PRESDIV | | | | | | | | RJW | | PSEG1 | | | PSEG2 | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | BOFF MSK | ERRM SK | CLKS RC | LPB | TWRN MSK | RWRN MSK | 0 | | SMP | BOFF REC | TSYN | LBUF | LOM | PROPSEG | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_CTRL1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**PRESDIV** | RW | Prescaler Division Factor<br>This 8-bit field defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The maximum value of this field is 0xFF, which gives a minimum Sclock frequency equal to the PE clock frequency divided by 256. See Protocol timing. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Sclock frequency = PE clock frequency / (PRESDIV + 1) |
| **23-22**<br>**RJW** | RW | Resync Jump Width<br>This 2-bit field defines the maximum number of time quanta that a bit time can be changed by one resynchronization. One time quantum is equal to the Sclock period. The valid programmable values are 0–3. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Resync Jump Width = RJW + 1. |
| **21-19**<br>**PSEG1** | RW | Phase Segment 1<br>This 3-bit field defines the length of phase segment 1 in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Phase Buffer Segment 1 = (PSEG1 + 1) × Time-Quanta. |
| **18-16**<br>**PSEG2** | RW | Phase Segment 2<br>This 3-bit field defines the length of phase segment 2 in the bit time. The valid programmable values are 1–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Phase Buffer Segment 2 = (PSEG2 + 1) × Time-Quanta. |

| | | |
|---|---|---|
| **15**<br>**BOFFMSK** | RW | Bus Off Interrupt Mask<br>This bit provides a mask for the Bus Off interrupt ESR1[BOFFINT].<br>0b - Bus Off interrupt disabled.<br>1b - Bus Off interrupt enabled. |
| **14**<br>**ERRMSK** | RW | Error Interrupt Mask<br>This bit provides a mask for the Error interrupt ESR1[ERRINT].<br>0b - Error interrupt disabled.<br>1b - Error interrupt enabled. |
| **13**<br>**CLKSRC** | RW | CAN Engine Clock Source<br>This bit selects the clock source to the CAN Protocol Engine (PE) to be either the peripheral clock or the oscillator clock. The selected clock is the one fed to the prescaler to generate the serial clock (Sclock). In order to guarantee reliable operation, this bit can be written only in Disable mode because it is blocked by hardware in other modes. See Protocol timing.<br>NOTE: The user must ensure the protocol engine clock tolerance according to the CAN Protocol standard (ISO 11898-1).<br>NOTE: See the clock distribution chapter to identify the proper clock source.<br>0b - The CAN engine clock source is the oscillator clock. Under this condition, the oscillator clock frequency must be lower than the bus clock.<br>1b - The CAN engine clock source is the peripheral clock. |
| **12**<br>**LPB** | RW | Loop Back Mode<br>This bit configures FLEXCAN to operate in Loop-Back mode. In this mode, FLEXCAN performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx FLEXCAN input pin is ignored and the Tx FLEXCAN output goes to the recessive state (logic 1). FLEXCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node.<br>In this mode, FLEXCAN ignores the bit sent during the ACK slot in the CAN Frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br>NOTE: In this mode, MCR[SRXDIS] cannot be asserted because this will impede the self-reception of a transmitted message.<br>NOTE: FDCTRL[TDCEN] and ETDC[ETDCEN] must be disabled when LPB is asserted.<br>0b - Loop Back disabled.<br>1b - Loop Back enabled. |

| | | |
|---|---|---|
| **11**<br>**TWRNMSK** | RW | Tx Warning Interrupt Mask<br>This bit provides a mask for the Tx Warning interrupt associated with the TWRNINT flag in the Error and Status Register 1 (ESR1). This bit is read as zero when MCR[WRNEN] is negated. This bit can be written only if MCR[WRNEN] is asserted.<br>0b - Tx Warning interrupt disabled.<br>1b - Tx Warning interrupt enabled. |
| **10**<br>**RWRNMSK** | RW | Rx Warning Interrupt Mask<br>This bit provides a mask for the Rx Warning interrupt associated with the RWRNINT flag in the Error and Status Register 1 (ESR1). This bit is read as zero when MCR[WRNEN] bit is negated. This bit can be written only if MCR[WRNEN] bit is asserted.<br>0b - Rx Warning interrupt disabled.<br>1b - Rx Warning interrupt enabled. |
| **7**<br>**SMP** | RW | CAN Bit Sampling<br>This bit defines the sampling mode of CAN bits at the Rx input. It can be written in Freeze mode only, because it is blocked by hardware in other modes.<br>NOTE: For proper operation, to assert SMP it is necessary to guarantee a minimum value of two TQs in CTRL1[PSEG1] (or CBT[EPSEG1]).This bit cannot be asserted when CAN FD is enabled (MCR[FDEN] = 1).<br>0b - Just one sample is used to determine the bit value.<br>1b - Three samples are used to determine the value of the received bit: the regular one (sample point) and two preceding samples; a majority rule is used. |

| | | |
|---|---|---|
| **6**<br>**BOFFREC** | RW | Bus Off Recovery<br>This bit defines how FLEXCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFFREC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FLEXCAN will resynchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFFREC bit can be reasserted again during Bus Off, but it will be effective only the next time the module enters Bus Off. If BOFFREC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery.<br>NOTE: See Bus off in the CAN Protocol standard (ISO 11898-1) for details.<br>0b - Automatic recovering from Bus Off state enabled.<br>1b - Automatic recovering from Bus Off state disabled. |
| **5**<br>**TSYN** | RW | Timer Sync<br>This bit enables a mechanism that resets the free running timer each time a message is received in message buffer 0. This feature provides means to synchronize multiple FLEXCAN stations with a special "SYNC" message, that is, global network time. If MCR[RFEN] is set (Legacy Rx FIFO enabled), the first available mailbox, according to CTRL2[RFFN] setting, is used for timer synchronization instead of MB0. This bit can be written in Freeze mode only because it is blocked by hardware in other modes.<br>0b - Timer sync feature disabled<br>1b - Timer sync feature enabled |
| **4**<br>**LBUF** | RW | Lowest Buffer Transmitted First<br>This bit defines the ordering mechanism for message buffer transmission. When asserted, MCR[LPRIOEN] does not affect the priority arbitration. This bit can be written in Freeze mode only, because it is blocked by hardware in other modes.<br>0b - Buffer with highest priority is transmitted first.<br>1b - Lowest number buffer is transmitted first. |

| | | |
|---|---|---|
| **3**<br>**LOM** | RW | Listen-Only Mode<br>This bit configures FLEXCAN to operate in Listen-Only mode. In this mode, transmission is disabled, all error counters described in the ECR register are frozen, and the module operates in a FLEXCAN Error Passive mode. Only messages acknowledged by another FLEXCAN station will be received. If FLEXCAN detects a message that has not been acknowledged, it will flag a BIT0 error without changing the receive error counter (ECR[RXERRCNT]), as if it was trying to acknowledge the message. Listen-Only mode is acknowledged by the state of ESR1[FLTCONF] indicating Passive Error. There can be some delay between the Listen-Only mode request and acknowledge.<br>This bit can be written in Freeze mode only because it is blocked by hardware in other modes.<br>0b - Listen-Only mode is deactivated.<br>1b - FLEXCAN module operates in Listen-Only mode. |
| **2-0**<br>**PROPSEG** | RW | Propagation Segment<br>This 3-bit field defines the length of the propagation segment in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Propagation segment time = (PROPSEG + 1) × time-quanta. Time-quantum = one Sclock period. |

### 60.8.3 FLEXCAN_TIMER (Free Running Timer)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TIMER | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_TIMER field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**TIMER** | RW | Timer Value<br>Contains the free-running counter value. |

### 60.8.4 FLEXCAN_RXMGMASK (Rx Mailboxes Global Mask Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MG | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MG | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_RXMGMASK field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**MG** | RW | Rx Mailboxes Global Mask Bits<br>These bits mask the mailbox filter bits. Note that the alignment with the ID word of the mailbox is not perfect as the two most significant MG bits affect the fields RTR and IDE, which are located in the Control and Status word of the mailbox. |

### 60.8.5 FLEXCAN_RX14MASK (Rx 14 Mask Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX14M | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX14M | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_RX14MASK field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-0**<br>**RX14M** | RW | Rx Buffer 14 Mask Bits<br>Each mask bit masks the corresponding mailbox 14 filter field in the same way that RXMGMASK masks other mailboxes' filters. See the description of the RXMGMASK register.<br>0b - The corresponding bit in the filter is "don't care."<br>1b - The corresponding bit in the filter is checked. |

### 60.8.6 FLEXCAN_RX15MASK (Rx 15 Mask Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX15M | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX15M | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_RX15MASK field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**RX15M** | RW | Rx Buffer 15 Mask Bits<br>Each mask bit masks the corresponding mailbox 15 filter field in the same way that RXMGMASK masks other mailboxes' filters. See the description of the RXMGMASK register.<br>0b - The corresponding bit in the filter is "don't care."<br>1b - The corresponding bit in the filter is checked. |

### 60.8.7 FLEXCAN_ECR (Error Counter Register)

Offset : 1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RXERRCNT_FAST | | | | | | | | TXERRCNT_FAST | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RXERRCNT | | | | | | | | TXERRCNT | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_ECR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**RXERRCNT_FAST** | RO | Receive Error Counter for fast bits<br>Receive error counter for errors detected in the data phase of received CAN FD messages with the BRS bit set. The RXERRCNT_FAST counter is read-only except in Freeze mode, where the CPU can write an 8-bit zero value only. |
| **23-16**<br>**TXERRCNT_FAST** | RO | Transmit Error Counter for fast bits<br>Transmit error counter for errors detected in the data phase of transmitted CAN FD messages with the BRS bit set. The TXERRCNT_FAST counter is read-only except in Freeze mode, where the CPU can write an 8-bit zero value only. |
| **15-8**<br>**RXERRCNT** | RO | Receive Error Counter<br>Receive error counter for all errors detected in received messages. The RXERRCNT counter is read-only except in Freeze mode, where it can be written by the CPU. |
| **7-0**<br>**TXERRCNT** | RO | Transmit Error Counter<br>Transmit error counter for all errors detected in transmitted messages. The TXERRCNT counter is read-only except in Freeze mode, where it can be written by the CPU. |

### 60.8.8 FLEXCAN_ESR1 (Error and Status 1 Register)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BIT1ERR_FAST | BIT0ERR_FAST | 0 | CRCERR_FAST | FRMERR_FAST | STFERR_FAST | 0 | | | | ERROVR | ERRINT_FAST | BOFFDONEINT | SYNCH | TWRNINT | RWRNINT |
| W | | | | | | | | | | | w1c | w1c | w1c | | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BIT1ERR | BIT0ERR | ACKERR | CRCERR | FRMERR | STFERR | TXWRN | RXWRN | IDLE | TX | FLTCONF | | RX | BOFFINT | ERRINT | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | w1c | w1c | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_ESR1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**BIT1ERR_FAST** | RO | Bit1 Error in the Data Phase of CAN FD frames with the BRS bit set<br>This bit indicates when an inconsistency occurs between the transmitted and the received bit in the data phase of CAN FD frames with the BRS bit set.<br>0b - No such occurrence.<br>1b - At least one bit sent as recessive is received as dominant. |
| **30**<br>**BIT0ERR_FAST** | RO | Bit0 Error in the Data Phase of CAN FD frames with the BRS bit set<br>This bit indicates when an inconsistency occurs between the transmitted and the received bit in the data phase of CAN FD frames with the BRS bit set.<br>0b - No such occurrence.<br>1b - At least one bit sent as dominant is received as recessive. |
| **28**<br>**CRCERR_FAST** | RO | Cyclic Redundancy Check Error in the CRC field of CAN FD frames with the BRS bit set<br>This bit indicates that a CRC error has been detected by the receiver node in the CRC field of CAN FD frames with the BRS bit set, that is, the calculated CRC is different from the received.<br>0b - No such occurrence.<br>1b - A CRC error occurred since last read of this register. |
| **27**<br>**FRMERR_FAST** | RO | Form Error in the Data Phase of CAN FD frames with the BRS bit set<br>This bit indicates that a form error has been detected by the receiver node in the data phase of CAN FD frames with the BRS bit set, that is, a fixed-form bit field contains at least one illegal bit.<br>0b - No such occurrence.<br>1b - A form error occurred since last read of this register. |
| **26**<br>**STFERR_FAST** | RO | Stuffing Error in the Data Phase of CAN FD frames with the BRS bit set<br>This bit indicates that a stuffing error has been detected in the data phase of CAN FD frames with the BRS bit set.<br>0b - No such occurrence.<br>1b - A stuffing error occurred since last read of this register. |

| | | |
|---|---|---|
| **21**<br>**ERROVR** | W1C | Error Overrun<br>This bit indicates that an error condition occurred when any error flag is already set. This bit is cleared by writing it to 1.<br>0b - Overrun has not occurred.<br>1b - Overrun has occurred. |
| **20**<br>**ERRINT_FAST** | W1C | Error interrupt for errors detected in Data Phase of CAN FD frames with BRS bit set<br>This bit indicates that at least one of the error bits detected in the data phase of CAN FD frames with the BRS bit set (BIT1ERR_FAST, BIT0ERR_FAST, CRCERR_FAST, FRMERR_FAST, or STFERR_FAST) is set. If the corresponding mask bit CTRL2[ERRMSK_FAST] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect.<br>0b - No such occurrence.<br>1b - Indicates setting of any error bit detected in the data phase of CAN FD frames with the BRS bit set. |
| **19**<br>**BOFFDONEINT** | W1C | Bus Off Done Interrupt<br>This bit is set when the Tx Error Counter (TXERRCNT) has finished counting 128 occurrences of 11 consecutive recessive bits on the FLEXCAN bus and is ready to leave Bus Off. If the corresponding mask bit in |
| **18**<br>**SYNCH** | RO | FLEXCAN Synchronization Status<br>This read-only flag indicates whether the FLEXCAN is synchronized to the FLEXCAN bus and able to participate in the communication process. It is set and cleared by the FLEXCAN. See the table in the overall ESR1 register description.<br>0b - FLEXCAN is not synchronized to the FLEXCAN bus.<br>1b - FLEXCAN is synchronized to the FLEXCAN bus. |
| **17**<br>**TWRNINT** | W1C | Tx Warning Interrupt Flag<br>If MCR[WRNEN] is asserted, TWRNINT is set when the TXWRN flag transitions from 0 to 1, meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control 1 Register (CTRL1[TWRNMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit.<br>Otherwise it will be set when WRNEN is set again. Writing 0 has no effect. This flag is not generated during Bus Off state. This bit is not updated during Freeze mode.<br>0b - No such occurrence.<br>1b - The Tx error counter transitioned from less than 96 to greater than or equal to 96. |

| | | |
|---|---|---|
| **16**<br>**RWRNINT** | W1C | Rx Warning Interrupt Flag<br>If MCR[WRNEN] is asserted, RWRNINT is set when the RXWRN flag transitions from 0 to 1, meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control 1 Register (CTRL1[RWRNMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit.<br>Otherwise it will be set when WRNEN is set again. Writing 0 has no effect. This bit is not updated during Freeze mode.<br>0b - No such occurrence.<br>1b - The Rx error counter transitioned from less than 96 to greater than or equal to 96. |
| **15**<br>**BIT1ERR** | RO | Bit1 Error<br>This bit indicates when an inconsistency occurs between the transmitted and the received bit in a non- CAN FD message or in the arbitration or data phase of a CAN FD message.<br>NOTE: This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits.<br>0b - No such occurrence.<br>1b - At least one bit sent as recessive is received as dominant. |
| **14**<br>**BIT0ERR** | RO | Bit0 Error<br>This bit indicates when an inconsistency occurs between the transmitted and the received bit in a non- CAN FD message or in the arbitration or data phase of a CAN FD message.<br>0b - No such occurrence.<br>1b - At least one bit sent as dominant is received as recessive. |
| **13**<br>**ACKERR** | RO | Acknowledge Error<br>This bit indicates that an Acknowledge Error has been detected by the transmitter node, that is, a dominant bit has not been detected during the ACK SLOT.<br>0b - No such occurrence.<br>1b - An ACK error occurred since last read of this register. |
| **12**<br>**CRCERR** | RO | Cyclic Redundancy Check Error |

| | | |
|---|---|---|
| **11**<br>**FRMERR** | RO | Form Error<br>This bit indicates that a form error has been detected in a non-FD message or else in an FD message's arbitration or data phase by the receiver node, that is, a fixed-form bit field contains at least one illegal bit.<br>0b - No such occurrence.<br>1b - A Form Error occurred since last read of this register. |
| **10**<br>**STFERR** | RO | Stuffing Error<br>This bit indicates that a stuffing error has been detected in a non-FD message or else in an FD message's arbitration or data phase by the receiver node.<br>0b - No such occurrence.<br>1b - A stuffing error occurred since last read of this register. |
| **9**<br>**TXWRN** | RO | TX Error Warning<br>This bit indicates when repetitive errors are occurring during message transmission and is affected by the value of TXERRCNT in ECR register only. This bit is not updated during Freeze mode.<br>0b - No such occurrence.<br>1b - TXERRCNT is greater than or equal to 96. |
| **8**<br>**RXWRN** | RO | Rx Error Warning<br>This bit indicates when repetitive errors are occurring during message reception and is affected by the value of RXERRCNT in ECR register only. This bit is not updated during Freeze mode.<br>0b - No such occurrence.<br>1b - RXERRCNT is greater than or equal to 96. |
| **7**<br>**IDLE** | RO | IDLE<br>This bit indicates when FLEXCAN bus is in IDLE state. See the table in the overall ESR1 register description.<br>0b - No such occurrence.<br>1b - FLEXCAN bus is now IDLE. |
| **6**<br>**TX** | RO | FLEXCAN In Transmission<br>This bit indicates if FLEXCAN is transmitting a message. See the table in the overall ESR1 register description.<br>0b - FLEXCAN is not transmitting a message.<br>1b - FLEXCAN is transmitting a message. |

| | | |
|---|---|---|
| **5-4**<br>**FLTCONF** | RO | Fault Confinement State<br>This 2-bit field indicates the confinement state of the FLEXCAN module. If CTRL1[LOM] is asserted, after a delay that depends on the CAN bit timing, ESR1[FLTCONF] will indicate Error Passive. The very same delay affects the way that ESR1[FLTCONF] reflects an update to ECR register by the CPU. It may be necessary to wait up to one CAN bit time to get them coherent again.<br>This bit field is affected by soft reset, but if the LOM bit is asserted, its reset value lasts just one CAN bit. After this time, ESR1[FLTCONF] reports Error Passive.<br>00b - Error Active<br>01b - Error Passive<br>1xb - Bus Off |
| **3**<br>**RX** | RO | This bit indicates if FLEXCAN is receiving a message. See the table in the overall ESR1 register description.<br>0b - FLEXCAN is not receiving a message.<br>1b - FLEXCAN is receiving a message. |
| **2**<br>**BOFFINT** | W1C | Bus Off Interrupt<br>This bit is set when FLEXCAN enters Bus Off state. If the corresponding mask bit in the Control Register 1 (CTRL1[BOFFMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1.<br>Writing 0 has no effect.<br>0b - No such occurrence.<br>1b - FLEXCAN module entered Bus Off state. |
| **1**<br>**ERRINT** | W1C | Error Interrupt<br>This bit indicates that at least one of the error bits (BIT1ERR, BIT0ERR, ACKERR, CRCERR, FRMERR, or STFERR) is set. If the corresponding mask bit CTRL1[ERRMSK] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect.<br>0b - No such occurrence.<br>1b - Indicates setting of any error bit in the Error and Status register. |

### 60.8.9 FLEXCAN_IMASK1 (Interrupt Masks 1 Register)

Offset : 28h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BUF31TO0M | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BUF31TO0M | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_IMASK1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**BUF31TO0M** | RW | Buffer MB i Mask<br>Each bit enables or disables the corresponding CAN message buffer interrupt for MB31 to MB0.<br>NOTE: Setting or clearing a bit in the IMASK1 Register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set.<br>0b - The corresponding buffer interrupt is disabled.<br>1b - The corresponding buffer interrupt is enabled. |

### 60.8.10 FLEXCAN_IFLAG1 (Interrupt Flags 1 Register)

Offset : 30h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BUF31TO8I | | | | | | | | | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BUF31TO8I | | | | | | | | BUF7I | BUF6I | BUF5I | BUF4TO1I | | | | BUF0I |
| W | w1c | | | | | | | | w1c | w1c | w1c | w1c | | | | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_IFLAG1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-8**<br>**BUF31TO8I** | W1C | Buffer MBi Interrupt<br>Each bit flags the corresponding CAN message buffer interrupt for MB31 to MB8.<br>0b - The corresponding buffer has no occurrence of successfully completed transmission or reception.<br>1b - The corresponding buffer has successfully completed transmission or reception. |

| | | |
|---|---|---|
| **7**<br>**BUF7I** | W1C | Buffer MB7 Interrupt Or Rx FIFO Overflow<br>When MCR[RFEN] is cleared (Rx FIFO disabled), this bit flags the interrupt for MB7.<br>NOTE: This flag is cleared by the FLEXCAN whenever MCR[RFEN] is changed by CPU writes.<br>The BUF7I flag represents Rx FIFO overflow when MCR[RFEN] is set. In this case, the flag indicates that a message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a mailbox.<br>0b - No occurrence of MB7 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO overflow when MCR[RFEN]=1<br>1b - MB7 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO overflow when MCR[RFEN]=1 |
| **6**<br>**BUF6I** | W1C | Buffer MB6 Interrupt Or Rx FIFO Warning<br>When MCR[RFEN] is cleared (Rx FIFO disabled), this bit flags the interrupt for MB6.<br>NOTE: This flag is cleared by the FLEXCAN whenever MCR[RFEN] is changed by CPU writes.<br>The BUF6I flag represents Rx FIFO warning when MCR[RFEN] is set. In this case, the flag indicates when the number of unread messages within the Rx FIFO is increased to five from four due to the reception of a new one, meaning that the Rx FIFO is almost full. Note that if the flag is cleared when the number of unread messages is greater than four, it does not assert again until the number of unread messages within the Rx FIFO is decreased to be equal to or less than four.<br>0b - No occurrence of MB6 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO almost full when MCR[RFEN]=1<br>1b - MB6 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO almost full when MCR[RFEN]=1 |

| | | |
|---|---|---|
| **5**<br>**BUF5I** | W1C | Buffer MB5 Interrupt Or Frames available in Rx FIFO<br>When MCR[RFEN] is cleared (Rx FIFO disabled), this field flags the interrupt for MB5.<br>NOTE: This flag is cleared by the FLEXCAN whenever MCR[RFEN] is changed by CPU writes.<br>When MCR[RFEN] is set (Rx FIFO enabled), the BUF5I flag represents "Frames available in Rx FIFO" and indicates that at least one frame is available to be read from the Rx FIFO. When the MCR[DMA] bit is enabled, this flag generates a DMA request and the CPU must not clear this bit by writing 1 in BUF5I.<br>0b - No occurrence of MB5 completing transmission/reception when MCR[RFEN]=0, or of frame(s) available in the FIFO, when MCR[RFEN]=1<br>1b - MB5 completed transmission/reception when MCR[RFEN]=0, or frame(s) available in the Rx FIFO when MCR[RFEN]=1. It generates a DMA request in case of MCR[RFEN] and MCR[DMA] are enabled. |
| **4-1**<br>**BUF4TO1I** | W1C | Buffer MB i Interrupt Or Reserved<br>When MCR[RFEN] is cleared (Rx FIFO disabled), these bits flag the interrupts for MB4 to MB1. NOTE: These flags are cleared by the FLEXCAN whenever MCR[RFEN] is changed by CPU writes. The BUF4TO1I flags are reserved when MCR[RFEN] is set.<br>0b - The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0.<br>1b - The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0. |
| **0**<br>**BUF0I** | W1C | Buffer MB0 Interrupt Or Clear FIFO bit<br>When MCR[RFEN] is cleared (Rx FIFO disabled), this field flags the interrupt for MB0. If the Rx FIFO is enabled, this bit is used to trigger the clear FIFO operation. This operation empties FIFO contents. Before performing this operation the CPU must service all FIFO related IFLAGs. When MCR[DMA] is enabled this operation also clears the BUF5I flag and consequently aborts the DMA request. The clear FIFO operation occurs when the CPU writes 1 in BUF0I. It is only allowed in Freeze mode and is blocked by hardware in other conditions.<br>0b - The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0.<br>1b - The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0. |

### 60.8.11 FLEXCAN_CTRL2 (Control 2 Register)

Offset : 34h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ERRM | BOFF | ECRW | WRMF | RFFN | | | | TASD | | | | | MRP | RRS | EACE |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | SK_FAST | DONEMSK | RE | RZ | | | | | | | | | | | | N |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | PREXCEN | BTE | ISOCANFDEN | EDFLTDIS | 0 | | | | | | | | | DIRECT_TRIG | DIRECT_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_CTRL2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**ERRMSK_FAST** | RW | Error Interrupt Mask for errors detected in the data phase of fast CAN FD frames<br>This bit provides a mask for the ERRINT_FAST interrupt in ESR1 register.<br>0b - ERRINT_FAST error interrupt disabled.<br>1b - ERRINT_FAST error interrupt enabled. |
| **30**<br>**BOFFDONEMSK** | RW | Bus Off Done Interrupt Mask<br>This bit provides a mask for the bus off done interrupt in ESR1 register.<br>0b - Bus off done interrupt disabled.<br>1b - Bus off done interrupt enabled. |
| **29**<br>**ECRWRE** | RW | Error-correction Configuration Register Write Enable<br>This bit enables the MECR register to be updated. This bit is automatically set to 0 if the protocol described in Detection and correction of memory errors is not followed.<br>0b - Disable update.<br>1b - Enable update. |
| **28**<br>**WRMFRZ** | RW | Write-Access To Memory In Freeze Mode<br>Enable unrestricted write access to FLEXCAN memory in Freeze mode. This bit can only be written in Freeze mode and has no effect out of Freeze mode.<br>MCR[RFEN] must not be set during FLEXCAN memory initialization.<br>0b - Maintain the write access restrictions.<br>1b - Enable unrestricted write access to FLEXCAN memory. |

| | | |
|---|---|---|
| **27-24**<br>**RFFN** | RW | Number Of Rx FIFO Filters<br>This 4-bit field defines the number of Rx FIFO filters. The maximum selectable number of filters is determined by the chip. This field can only be written in Freeze mode as it is blocked by hardware in other modes. This field must not be programmed with values that cause the number of message buffers occupied by Legacy Rx FIFO and Legacy Rx FIFO ID Filter to exceed the number of mailboxes present, as defined by MCR[MAXMB].<br>NOTE: Each group of eight filters occupies a memory space equivalent to two message buffers, which means that the more filters are implemented the fewer mailboxes will be available.<br>Considering that the Legacy Rx FIFO occupies the memory space originally reserved for MB0-5, RFFN should be programmed with a value corresponding to a number of filters not greater than the number of available memory words, which can be calculated as follows:<br>(SETUP_MB - 6) × 4<br>where SETUP_MB is the smaller of the parameter NUMBER_OF_MB and MCR[MAXMB]. The number of remaining mailboxes available will be:<br>(SETUP_MB - 8) - (RFFN × 2)<br>If the number of Legacy Rx FIFO filters programmed through RFFN exceeds the SETUP_MB value (memory space available), then the exceeding ones will not be functional.<br>NOTE:<br>The number of the last remaining available mailboxes is defined by the smaller of NUMBER_OF_MB minus 1 and MCR[MAXMB].<br>If Rx Individual Mask registers are not enabled then all Legacy Rx FIFO filters are affected by the Legacy Rx FIFO Global Mask. |
| **23-19**<br>**TASD** | RW | Tx Arbitration Start Delay<br>This 5-bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on FLEXCAN bus. See Tx arbitration start delay for more details. This field can be written only in Freeze mode because it is blocked by hardware in other modes. |
| **18**<br>**MRP** | RW | Mailboxes Reception Priority<br>If this bit is set the matching process starts from the mailboxes and if no match occurs the matching continues on the Legacy Rx FIFO or Enhanced RX FIFO . This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br>NOTE: The descriptions of the field settings vary by module instance. |

| | | |
|---|---|---|
| **17**<br>**RRS** | RW | Remote Request Storing<br>If this bit is asserted a remote request frame is submitted to a matching process and stored in the corresponding message buffer in the same fashion as a data frame. No automatic remote response frame will be generated.<br>If this bit is negated the remote request frame is submitted to a matching process and an automatic remote response frame is generated if a message buffer with CODE=0b1010 is found with the same ID.<br>This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br>0b - Remote response frame is generated.<br>1b - Remote request frame is stored. |
| **16**<br>**EACEN** | RW | Entire Frame Arbitration Field Comparison Enable For Rx Mailboxes<br>This bit controls the comparison of IDE and RTR bits within Rx mailbox filters with their corresponding bits in the incoming frame by the matching process. This bit does not affect matching for Legacy Rx FIFO or Enhanced Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.<br>0b - Rx mailbox filter's IDE bit is always compared and RTR is never compared despite mask bits.<br>1b - Enables the comparison of both Rx mailbox filter's IDE and RTR bit with their corresponding bits within the incoming frame. Mask bits do apply. |
| **14**<br>**PREXCEN** | RW | Protocol Exception Enable<br>This bit enables the protocol exception feature. This field is writable only in Freeze mode.<br>NOTE: See Protocol exception event in the CAN Protocol standard (ISO 11898-1) for details.<br>0b - Protocol exception is disabled.<br>1b - Protocol exception is enabled. |

| | | |
|---|---|---|
| **13**<br>**BTE** | RW | Bit Timing Expansion enable<br>This field enables the use of the EPRS, EDCBT and ENCBT registers to configure the CAN bit timing segments, instead of using the bit timing fields of CBT, FDCBT, and CTRL1 registers.<br>If CTRL2[BTE] is set:<br>PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW fields in CTRL1 register are read as zero, and a write operation in them has no effect.<br>EPRESDIV, ERJW, EPROPSEG, EPSEG1, and EPSEG2 fields in the CBT register, and the FDCBT register, are read as zero and a write operation to them has no effect.<br>The ETDCOFF, ETDCEN, ETDCFAIL, and ETDCVAL fields of the ETDC register are used by FLEXCAN instead of TDCOFF, TDCEN, TDCFAIL, and TDCVAL fields in FDCTRL register, which are read as zero, and a write operation in them has no effect.<br>ETDC[TDMDIS] can be used for disabling transceiver delay measurement.<br><br>NOTE: This field is not supported in every instance.<br>0b - CAN Bit timing expansion is disabled.<br>1b - CAN bit timing expansion is enabled. |
| **12**<br>**ISOCANFDEN** | RW | ISO CAN FD Enable<br>This field enables the CAN FD protocol according to ISO specification (ISO 11898-1) (see CAN FD ISO compliance).<br>This field is writable only in Freeze mode.<br>NOTE: FLEXCAN is able to transmit FD frame format according to CAN Protocol standard (ISO 11898-1).<br>0b - FLEXCAN operates using the non-ISO CAN FD protocol.<br>1b - FLEXCAN operates using the ISO CAN FD protocol (ISO 11898-1). |
| **11**<br>**EDFLTDIS** | RW | Edge Filter Disable<br>This bit disables the edge filter used during the bus integration state. When the Edge Filter is enabled, two consecutive nominal time quanta with dominant bus state are required to detect an edge that causes synchronization. When synchronization occurs, the counting of the sequence of eleven consecutive recessive bits is restarted. The edge filter prevents the dominant pulses that are shorter than a nominal bit time (present during the data phase of an FD frame) from being mistaken for an idle condition.<br>This field is writable only in Freeze mode.<br>NOTE: See Bus Integration state in the CAN Protocol standard (ISO 11898-1) for details.<br>0b - Edge filter is enabled<br>1b - Edge filter is disabled |

| | | |
|---|---|---|
| **1**<br>**DIRECT_TRIG** | RW | Direct Connection Select for CAN Rx<br>If DIRECT_EN is set, this bit is used to select the direct CAN connection through pads TRGSEL_IN0 - 15.<br>0b - Does not select a TRGSEL_IN source from pad for direct CAN connection<br>1b - Select a TRGSEL_IN source from pad for direct CAN connection<br>Note: To select one pad from TRGSEL_IN0 - 15, configure the register SCM[FLEXCAN_ROUTING]. |
| **0**<br>**DIRECT_EN** | RW | Enable Direct Connection for CAN Tx and Rx<br>This bit enables the direct CAN TX/RX connection without CAN PHY function.<br>0b - Disable direct CAN connection function<br>1b - Enable direct CAN connection function |

### 60.8.12 FLEXCAN_ESR2 (Error and Status 2 Register)

Offset : 38h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | LPTM | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | VPS | IMB | 0 | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_ESR2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **22-16**<br>**LPTM** | RO | Lowest Priority Tx Mailbox<br>If ESR2[VPS] is asserted, this field indicates the lowest number inactive mailbox (see the ESR2[IMB] bit description). If there is no inactive mailbox then the mailbox indicated depends on the value of CTRL1[LBUF]. If CTRL1[LBUF] is negated, then the mailbox indicated is the one that has the greatest arbitration value (see Highest-priority mailbox first). If CTRL1[LBUF] is asserted then the mailbox indicated is the highest number active Tx mailbox. If a Tx mailbox is being transmitted it is not considered in LPTM calculation. If ESR2[IMB] is not asserted and a frame is transmitted successfully, LPTM is updated with its mailbox number. |

| | | |
|---|---|---|
| **14**<br>**VPS** | RO | Valid Priority Status<br>This bit indicates whether ESR2[IMB] and ESR2[LPTM] contents are currently valid or not. It is asserted upon every complete Tx arbitration process unless the CPU writes to Control and Status word of a mailbox that has already been scanned, that is, it is behind Tx Arbitration Pointer, during the Tx arbitration process. If there is no inactive mailbox and only one Tx mailbox that is being transmitted then VPS is not asserted. This bit is negated upon the start of every Tx arbitration process or upon a write to Control and Status word of any mailbox.<br>NOTE: ESR2[VPS] is not affected by any CPU write into Control Status (C/S) of an MB that is blocked by abort mechanism. When MCR[AEN] is asserted, the abort code write in C/S of an MB that is being transmitted (pending abort), or any write attempt into a Tx MB with IFLAG set is blocked.<br>0b - Contents of IMB and LPTM are invalid.<br>1b - Contents of IMB and LPTM are valid. |
| **13**<br>**IMB** | RO | Inactive Mailbox<br>If ESR2[VPS] is asserted, this bit indicates whether there is any inactive mailbox (CODE field is either 0b1000 or 0b0000). This bit is asserted in the following cases:<br>During arbitration, if an ESR2[LPTM] is found and it is inactive.<br>If ESR2[IMB] is not asserted and a frame is transmitted successfully.<br><br>This bit is always cleared in start of arbitration (see Arbitration process).<br>NOTE: ESR2[LPTM] mechanism has the following behavior: if an MB is successfully transmitted and ESR2[IMB]=0 (no inactive mailbox), then ESR2[VPS] and ESR2[IMB] are asserted and the index related to the MB just transmitted is loaded into ESR2[LPTM].<br>0b - If ESR2[VPS] is asserted, the ESR2[LPTM] is not an inactive mailbox.<br>1b - If ESR2[VPS] is asserted, there is at least one inactive mailbox. LPTM content is the number of the first one. |

### 60.8.13 FLEXCAN_CRCR (CRC Register)

Offset : 44h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | MBCRC | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | TXCRC | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_CRCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **22-16**<br>**MBCRC** | RO | CRC Mailbox<br>This field indicates the number of the mailbox corresponding to the value in CRCR[TXCRC] field. |
| **14-0**<br>**TXCRC** | RO | Transmitted CRC value<br>This field indicates the CRC value of the last transmitted message for non-FD frames. For FD frames, CRC value is reported in FDCRC register. |

### 60.8.14 FLEXCAN_RXFGMASK (Legacy Rx FIFO Global Mask Register)

If Legacy Rx FIFO is enabled, RXFGMASK is used to mask the Legacy Rx FIFO ID filter table elements that do not have a corresponding RXIMR according to CTRL2[RFFN] field setting.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Offset : 48h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FGM | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FGM | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_RXFGMASK field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**FGM** | RW | Legacy Rx FIFO Global Mask Bits<br>These bits mask the ID filter table elements bits in a perfect alignment.<br>0b - The corresponding bit in the filter is "don't care." |

### 60.8.15 FLEXCAN_RXFIR (Legacy Rx FIFO Information Register)

Offset : 4Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | IDHIT | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_RXFIR field descriptions**

| Field | Type | Description |
|---|---|---|
| **8-0**<br>**IDHIT** | RO | Identifier Acceptance Filter Hit Indicator<br>This field indicates which Identifier Acceptance filter was hit by the received message that is in the output of the Legacy Rx FIFO. If multiple filters match the incoming message ID then the first matching IDAF found (lowest number) by the matching process is indicated. This field is valid only when IFLAG1[BUF5I] is asserted. |

### 60.8.16 FLEXCAN_CBT (CAN Bit Timing Register)

Offset : 50h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BTF | | | EPRESDIV | | | | | | | | ERJW | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EPROPSEG | | | | | | EPSEG1 | | | | | EPSEG2 | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_CBT field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31**<br>**BTF** | RW | Bit Timing Format Enable<br>Enables the use of extended CAN bit timing fields EPRESDIV, EPROPSEG, EPSEG1, EPSEG2. and ERJW replacing the CAN bit timing variables defined in CTRL1 register. This field can be written in Freeze mode only.<br>0b - Extended bit time definitions disabled.<br>1b - Extended bit time definitions enabled. |
| **30-21**<br>**EPRESDIV** | RW | Extended Prescaler Division Factor<br>This 10-bit field defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency when CBT[BTF] is asserted, otherwise it has no effect. It extends the CTRL1[PRESDIV] value range.<br>The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency (see Protocol timing). This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Sclock frequency = PE clock frequency / (EPRESDIV + 1) |
| **20-16**<br>**ERJW** | RW | Extended Resync Jump Width<br>This 5-bit field defines the maximum number of time quanta that a bit time can be changed by one re- synchronization when CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CTRL1[RJW] value range.<br>One time quantum is equal to the Sclock period. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Resync Jump Width = ERJW + 1. |
| **15-10**<br>**EPROPSEG** | RW | Extended Propagation Segment<br>This 6-bit field defines the length of the propagation segment in the bit time when CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CTRL1[PROPSEG] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Propagation Segment Time = (EPROPSEG + 1) × Time-Quanta.<br>Time-Quantum = one Sclock period. |
| **9-5**<br>**EPSEG1** | RW | Extended Phase Segment 1<br>This 5-bit field defines the length of phase segment 1 in the bit time when CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CTRL1[PSEG1] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Phase Buffer Segment 1 = (EPSEG1 + 1) × Time-Quanta.<br>Time-Quantum = one Sclock period. |

| | | |
|---|---|---|
| **4-0**<br>**EPSEG2** | RW | Extended Phase Segment 2<br>This 5-bit field defines the length of phase segment 2 in the bit time when CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CTRL1[PSEG2] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Phase Buffer Segment 1 = (EPSEG2 + 1) × Time-Quanta.<br>Time-Quantum = one Sclock period. |

### 60.8.17 FLEXCAN_RAMn (Message Buffer Registers)

Offset : 80h+N*4h (N : 0 ~ 127)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_RAMn field descriptions**

| Field | Type | Description |
|---|---|---|

### 60.8.18 FLEXCAN_RXIMRn (Rx Individual Mask registers)

Offset : 880h+N*4h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MI | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MI | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_RXIMRn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0<br>MI** | RW | Individual Mask Bits<br>Each individual mask bit masks the corresponding bit in both the mailbox filter and Legacy Rx FIFO ID filter table element in distinct ways.<br>For mailbox filters, see the RXMGMASK register description.<br>For Legacy Rx FIFO ID filter table elements, see the RXFGMASK register description.<br>0b - The corresponding bit in the filter is "don't care."<br>1b - The corresponding bit in the filter is checked. |

### 60.8.19 FLEXCAN_MECR (Memory Error Control Register)

Offset : AE0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ECRWRDIS | 0 | | | | | | | | | | | HANCEI_MSK | FANCEI_MSK | 0 | CEI_MSK |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | HAERRIE | FAERRIE | EXTERRIE | 0 | | | RERRDIS | ECCDIS | NCEFAFRZ | 0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_MECR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31<br>ECRWRDIS** | RW | Error Configuration Register Write Disable Disables writes on this register.<br>This bit is automatically set to 1 (disabled) when CTRL2[ECRWRE] is enabled. The protocol described in<br>section Detection and correction of memory errors must be followed.<br>0b - Write is enabled.<br>1b - Write is disabled. |
| **19<br>HANCEI_MSK** | RW | Host Access With Non-Correctable Errors Interrupt Mask<br>Enables the interrupt in case of non-correctable errors detected in memory reads issued by the host (CPU).<br>0b - Interrupt is disabled.<br>1b - Interrupt is enabled. |

| | | |
|---|---|---|
| **18**<br>**FANCEI_MSK** | RW | FLEXCAN Access With Non-Correctable Errors Interrupt Mask<br>Enables the interrupt in case of non-correctable errors detected in memory reads issued by the FLEXCAN internal processes.<br>0b - Interrupt is disabled.<br>1b - Interrupt is enabled. |
| **16**<br>**CEI_MSK** | RW | Correctable Errors Interrupt Mask<br>Enables the interrupt in case of correctable errors detected in memory reads issued by the host or FLEXCAN internal processes.<br>0b - Interrupt is disabled.<br>1b - Interrupt is enabled. |
| **15**<br>**HAERRIE** | RW | Host Access Error Injection Enable<br>Enables the injection of errors only in memory reads issued by the host (CPU).<br>0b - Injection is disabled.<br>1b - Injection is enabled. |
| **14**<br>**FAERRIE** | RW | FLEXCAN Access Error Injection Enable<br>Enables the injection of errors only in memory reads issued by the FLEXCAN internal processes.<br>0b - Injection is disabled.<br>1b - Injection is enabled. |
| **13**<br>**EXTERRIE** | RW | Extended Error Injection Enable<br>Memory accesses performed by internal FLEXCAN processes are 64-bit. This bit extends the error injection on 32-bit memory accesses to the complementary 32-bit word using the same 32-bit error injection data and parity words. See Error Injection Data Pattern register (ERRIDPR) and Error Injection Parity Pattern register (ERRIPPR).<br>0b - Error injection is applied only to the 32-bit word.<br>1b - Error injection is applied to the 64-bit word. |
| **9**<br>**RERRDIS** | RW | Error Report Disable<br>Disables the update of the error report registers. The update of error-related flags and the generation of bus transfer errors are still active.<br>NOTE: When reading the report registers, this bit must be set to assure coherence on the consecutive register reads.<br>0b - Enable updates of the error report registers.<br>1b - Disable updates of the error report registers. |

| | | |
|---|---|---|
| **8**<br>**ECCDIS** | RW | Error Correction Disable<br>Disables completely the memory detection and correction mechanism. Besides disabling the error report mechanism, it also stops the update of the error-related flags and generation of bus transfer errors. The parity bits continue being calculated and written into memory on write transactions.<br>0b - Enable memory error correction.<br>1b - Disable memory error correction. |
| **7**<br>**NCEFAFRZ** | RW | Non-Correctable Errors In FLEXCAN Access Put Device In Freeze Mode<br>Determines the response when a non-correctable error is detected in a memory read performed by FLEXCAN internal processes. In this case, entering Freeze mode prevents corrupted data from being treated as valid by FLEXCAN internal processes.<br>0b - Keep normal operation.<br>1b - Put FLEXCAN in Freeze mode (see section "Freeze mode"). |

### 60.8.20 FLEXCAN_ERRIAR (Error Injection Address Register)

Offset : AE4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | INJADDR_H | | | | | | | INJADDR_L | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_ERRIAR field descriptions**

| Field | Type | Description |
|---|---|---|
| **13-2**<br>**INJADDR_H** | RW | Error Injection Address High<br>This read-write field defines the twelve most significant bits of the physical RAM address where error is to be injected. |
| **1-0**<br>**INJADDR_L** | RO | Error Injection Address Low<br>This read-only field defines the two least significant bits of the physical RAM address where error is to be injected. They ensure that the address is on a thirty-two-bit boundary. |

### 60.8.21 FLEXCAN_ERRIDPR (Error Injection Data Pattern Register)

Offset : AE8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DFLIP | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DFLIP | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_ERRIDPR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**DFLIP** | RW | Data Flip Pattern<br>Bits set to 1 in the flip pattern cause the corresponding data bit in the word read from memory to invert. |

### 60.8.22 FLEXCAN_ERRIPPR (Error Injection Parity Pattern Register)

Offset : AECh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | | | PFLIP3 | | | | | 0 | | | PFLIP2 | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | | | PFLIP1 | | | | | 0 | | | PFLIP0 | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_ERRIPPR field descriptions**

| Field | Type | Description |
|---|---|---|
| **28-24**<br>**PFLIP3** | RW | Parity Flip Pattern For Byte 3 (most significant) |

| | | |
|---|---|---|
| **20-16**<br>**PFLIP2** | RW | Parity Flip Pattern For Byte 2 |
| **12-8**<br>**PFLIP1** | RW | Parity Flip Pattern For Byte 1 |
| **4-0**<br>**PFLIP0** | RW | Parity Flip Pattern For Byte 0 (Least Significant) |

### 60.8.23 FLEXCAN_RERRAR (Error Report Address Register)

Offset : AF0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | NCE | 0 | | | | | SAID | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | ERRADDR | | | | | | | | | | | | 0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_RERRAR field descriptions**

| Field | Type | Description |
|---|---|---|
| **24**<br>**NCE** | RO | Non-Correctable Error<br>Indicates that the report is due to a non-correctable error.<br>0b - Reporting a correctable error<br>1b - Reporting a non-correctable error |
| **18-16**<br>**SAID** | RO | SAID[2] — Identification of the requester of the memory read request:<br>0 = Requested by FLEXCAN internal processes<br>1 = Requested by host (CPU)<br><br>SAID[1] — Details of FLEXCAN operation:<br>0 = Move<br>1 = Scanning<br><br>SAID[0] — Operation that requested the memory read:<br>0 = Transmission<br>1 = Reception |

| | | |
|---|---|---|
| **13-2**<br>**ERRADDR** | RO | Address Where Error Detected<br>See the description of the Error Injection Address Register (ERRIAR). |

### 60.8.24 FLEXCAN_RERRDR (Error Report Data Register)

Offset : AF4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RDATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RDATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_RERRDR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**RDATA** | RO | Raw data word read from memory with error |

### 60.8.25 FLEXCAN_RERRSYNR (Error Report Syndrome Register)

Offset : AF8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BE3 | 0 | | SYND3 | | | | | BE2 | 0 | | SYND2 | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BE1 | 0 | | SYND1 | | | | | BE0 | 0 | | SYND0 | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_RERRSYNR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31**<br>**BE3** | RO | Byte Enabled For Byte 3 (most significant)<br>0b - The byte was not read.<br>1b - The byte was read. |
| **28-24**<br>**SYND3** | RO | Error Syndrome For Byte 3 (most significant) |
| **23**<br>**BE2** | RO | Byte Enabled For Byte 2<br>0b - The byte was not read.<br>1b - The byte was read. |
| **20-16**<br>**SYND2** | RO | Error Syndrome For Byte 2 |
| **15**<br>**BE1** | RO | Byte Enabled For Byte 1<br>0b - The byte was not read.<br>1b - The byte was read. |
| **12-8**<br>**SYND1** | RO | Error Syndrome for Byte 1 |
| **7**<br>**BE0** | RO | Byte Enabled For Byte 0 (least significant)<br>0b - The byte was not read.<br>1b - The byte was read. |
| **4-0**<br>**SYND0** | RO | Error Syndrome For Byte 0 (least significant) |

### 60.8.26 FLEXCAN_ERRSR (Error Status Register)

Offset : AFCh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | HANCEIF | FANCEIF | 0 | CEIF |
| W | | | | | | | | | | | | | w1c | w1c | | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | HANCEIOF | FANCEIOF | 0 | CEIOF |
| W | | | | | | | | | | | | | w1c | w1c | | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_ERRSR field descriptions**

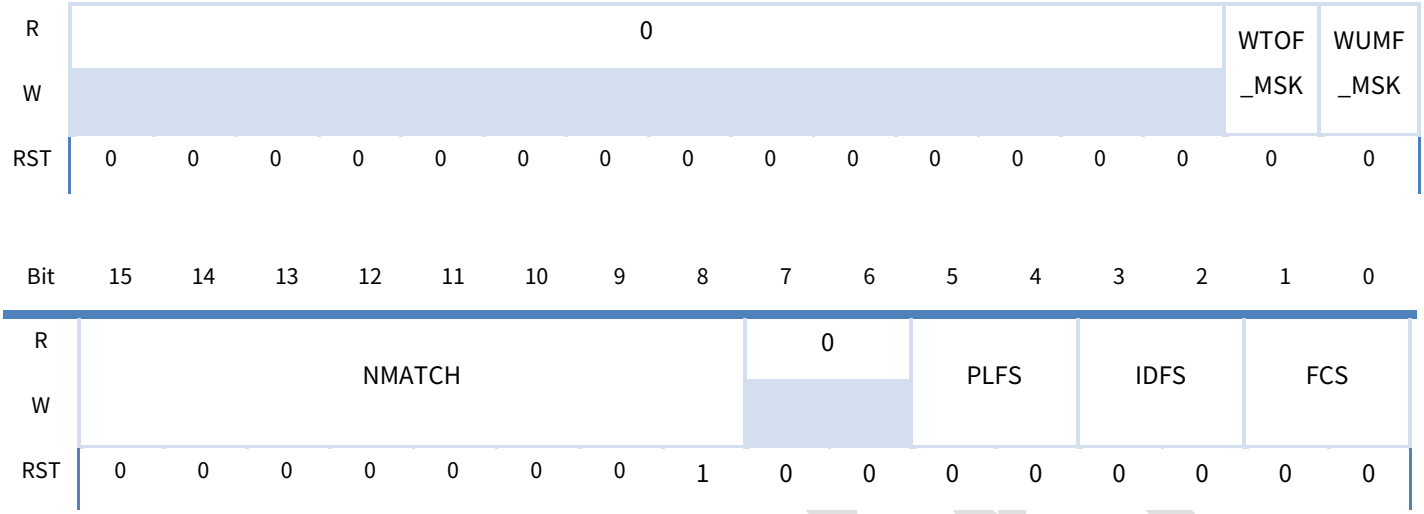
| Field | Type | Description |
|---|---|---|
| **19**<br>**HANCEIF** | W1C | Host Access With Non-Correctable Error Interrupt Flag<br>Indicates that a non-correctable error was detected in a memory read initiated by host. A bus transfer error is asserted for that access. If MECR[HANCEI_MSK] is set, the interrupt is asserted.<br>0b - No non-correctable errors were detected in host accesses so far.<br>1b - A non-correctable error was detected in a host access. |
| **18**<br>**FANCEIF** | W1C | FLEXCAN Access With Non-Correctable Error Interrupt Flag<br>Indicates that a non-correctable error was detected in a memory read initiated by FLEXCAN internal processes. If MECR[FANCEI_MSK] is set, the interrupt is asserted.<br>0b - No non-correctable errors were detected in FLEXCAN accesses so far.<br>1b - A non-correctable error was detected in a FLEXCAN access. |
| **16**<br>**CEIF** | W1C | Correctable Error Interrupt Flag<br>Indicates that a correctable error was detected in a memory read. If MECR[CEI_MSK] is set, the interrupt is asserted.<br>0b - No correctable errors were detected so far.<br>1b - A correctable error was detected. |
| **3**<br>**HANCEIOF** | W1C | Host Access With Non-Correctable Error Interrupt Overrun Flag<br>Indicates that a non-correctable error was detected in a memory read initiated by host when HANCEIF was set. No interrupt is associated with this flag. See Error indication.<br>0b - No overrun on non-correctable errors in host access<br>1b - Overrun on non-correctable errors in host access |
| **2**<br>**FANCEIOF** | W1C | FLEXCAN Access With Non-Correctable Error Interrupt Overrun Flag<br>Indicates that a non-correctable error was detected in a memory read initiated by FLEXCAN internal processes when FANCEIF was set. No interrupt is associated to this flag. See Error indication.<br>0b - No overrun on non-correctable errors in FLEXCAN access<br>1b - Overrun on non-correctable errors in FLEXCAN access |
| **0**<br>**CEIOF** | W1C | Correctable Error Interrupt Overrun Flag<br>Indicates that a correctable error was detected in a memory read when CEIF was set. No interrupt is associated to this flag. See Error indication.<br>0b - No overrun on correctable errors<br>1b - Overrun on correctable errors |

### 60.8.27 FLEXCAN_CTRL1_PN (Pretended Networking Control 1 Register)

Note: This register is for FLEXCAN0, FLEXCAN1, and FLEXCAN2 only.

Offset : B00h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|



| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | WTOF_MSK | WUMF_MSK |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | NMATCH | | | | | | | | 0 | | PLFS | | IDFS | | FCS | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_CTRL1_PN field descriptions**

| Field | Type | Description |
|---|---|---|
| **17**<br>**WTOF_MSK** | RW | Wake Up by Timeout Flag Mask Bit<br>This bit masks the generation of a wakeup event originated by a timeout.<br>0b - Timeout wakeup event is disabled<br>1b - Timeout wakeup event is enabled |
| **16**<br>**WUMF_MSK** | RW | Wake Up by Match Flag Mask Bit<br>This bit masks the generation of a wakeup event originated by a successful filtered Rx message.<br>0b - Wakeup match event is disabled<br>1b - Wakeup match event is enabled |
| **15-8**<br>**NMATCH** | RW | Number of Messages Matching the Same Filtering Criteria<br>This 8-bit field defines the number of times a given message must match the predefined filtering criteria for ID and/or PL before generating a wakeup event. This quantity can be configured in the 1 to 255 range by using values from 0x01 to 0xFF, respectively.<br>00000001b - Received message must match the predefined filtering criteria for ID and/or PL once before generating a wakeup event.<br>00000010b - Received message must match the predefined filtering criteria for ID and/or PL twice before generating a wakeup event.<br>11111111b - Received message must match the predefined filtering criteria for ID and/or PL 255 times before generating a wakeup event. |

| | | |
|---|---|---|
| **5-4**<br>**PLFS** | RW | Payload Filtering Selection<br>This 2-bit field selects the level of payload filtering to be applied when FLEXCAN is under Pretended Networking mode. Filtering does not accept remote messages (RTR=1) when payload filtering is active.<br>00b - Match upon a payload contents against an exact target value<br>01b - Match upon a payload value greater than or equal to a specified target value<br>10b - Match upon a payload value smaller than or equal to a specified target value<br>11b - Match upon a payload value inside a range, greater than or equal to a specified lower limit, and smaller than or equal to a specified upper limit |
| **3-2**<br>**IDFS** | RW | ID Filtering Selection<br>This 2-bit field selects the level of ID filtering to be applied when FLEXCAN is under Pretended Networking mode. In ID filtering, the IDE and RTR bits are also considered as part of the reception filter if IDE_MSK and RTR_MSK bits in the FLT_ID2_IDMASK register are set.<br>00b - Match upon ID contents against an exact target value<br>01b - Match upon an ID value greater than or equal to a specified target value 10b - Match upon an ID value smaller than or equal to a specified target value<br>11b - Match upon an ID value inside a range, greater than or equal to a specified lower limit, and smaller than or equal to a specified upper limit |
| **1-0**<br>**FCS** | RW | Filtering Combination Selection<br>This 2-bit field selects the filtering criteria to be applied when FLEXCAN is under Pretended Networking mode. See Receive process under Pretended Networking mode for more details.<br>00b - Message ID filtering only<br>01b - Message ID filtering and payload filtering<br>10b - Message ID filtering occurring a specified number of times<br>11b - Message ID filtering and payload filtering a specified number of times |

### 60.8.28 FLEXCAN_CTRL2_PN (Pretended Networking Control 2 Register)

Offset : B04h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCHTO | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_CTRL2_PN field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-0**<br>**MATCHTO** | RW | Timeout for No Message Matching the Filtering Criteria<br>This 16-bit field defines a timeout value that generates a wakeup event if MCR[PNET_EN] is asserted. If the timeout counter reaches the target value, when FLEXCAN is under Pretended Networking mode, then a wakeup event is generated. The timeout limit can be configured from 1 to 65535 to control an internal16-bit up-count timer to produce a trigger upon reaching this configured value. The internal timer is incremented based on periodic time ticks, which period is 64 times the CAN Bit Time unit. When MATCHTO is configured with 0x0000 the timeout is disabled. |

### 60.8.29 FLEXCAN_WU_MTC (Pretended Networking Wake Up Match Register)

Offset : B08h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | WTOF | WUMF |
| W | | | | | | | | | | | | | | | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MCOUNTER | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_WU_MTC field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **17**<br>**WTOF** | W1C | Wake Up by Timeout Flag Bit<br>This bit identifies whether the FLEXCAN has detected a timeout event during a time interval defined by CTRL2_PN[MATCHTO]. This flag generates a wakeup event if CTRL1_PN[WTOF_MSK] is enabled.<br>0b - No wakeup by timeout event detected<br>1b - Wakeup by timeout event detected |
| **16**<br>**WUMF** | W1C | Wake Up by Match Flag Bit<br>This bit identifies whether the FLEXCAN has detected a matching Rx incoming message, which passed the filtering criteria specified in CTRL1_PN register. This flag generates a wakeup event if CTRL1_PN[WUMF_MSK] is enabled.<br>0b - No wakeup by match event detected<br>1b - Wakeup by match event detected |
| **15-8**<br>**MCOUNTER** | RO | Number of Matches when in Pretended Networking<br>This 8-bit field reports the number of times a given message has matched the predefined filtering criteria for ID and/or PL before a wakeup event. This register is reset by FLEXCAN when it enters in Pretended Networking mode, and is not affected by soft reset. |

### 60.8.30 FLEXCAN_FLT_ID1 (Pretended Networking ID Filter 1 Register)

Offset : B0Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | FLT_ | FLT_ | FLT_ID1 | | | | | | | | | | | | |
| W | | IDE | RTR | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FLT_ID1 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_FLT_ID1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **30**<br>**FLT_IDE** | RW | ID Extended Filter<br>This bit identifies whether the frame format is standard or extended. It is used as part of the ID reception filter.<br>0b - Accept standard frame format<br>1b - Accept extended frame format |

| | | |
|---|---|---|
| **29**<br>**FLT_RTR** | RW | Remote Transmission Request Filter<br>This bit identifies whether the frame is remote or not. It is used as part of the ID reception filter.<br>0b - Reject remote frame (accept data frame)<br>1b - Accept remote frame |
| **28-0**<br>**FLT_ID1** | RW | ID Filter 1 for Pretended Networking filtering<br>This 29-bit field defines either the 29 bits of an extended frame format, considering all bits, or the 11 bits of a standard frame format, considering just the 11 leftmost bits. |

### 60.8.31 FLEXCAN_FLT_DLC (Pretended Networking DLC Filter Register)

Offset : B10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | FLT_DLC_LO | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | FLT_DLC_HI | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

**FLEXCAN_FLT_DLC field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16**<br>**FLT_DLC_LO** | RW | Lower Limit for Length of Data Bytes Filter<br>This field specifies the lower limit for the number of data bytes considered valid for payload comparison. It is used as part of payload reception filter. |
| **3-0**<br>**FLT_DLC_HI** | RW | Upper Limit for Length of Data Bytes Filter<br>This field specifies the upper limit for the number of data bytes considered valid for payload comparison. It is used as part of payload reception filter. |

### 60.8.32 FLEXCAN_PL1_LO (Pretended Networking Payload Low Filter 1 Register)

Offset : B14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA_BYTE_0 | | | | | | | | DATA_BYTE_1 | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | DATA_BYTE_2 | | | | | | | | DATA_BYTE_3 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_PL1_LO field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24 DATA_BYTE_0** | RW | Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 0. |
| **23-16 DATA_BYTE_1** | RW | Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 1. |
| **15-8 DATA_BYTE_2** | RW | Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 2. |
| **7-0 DATA_BYTE_3** | RW | Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 3. |

### 60.8.33 FLEXCAN_PL1_HI (Pretended Networking Payload High Filter 1 Register)

Offset : B18h

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| R | DATA_BYTE_4 | | | | | | | | DATA_BYTE_5 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | DATA_BYTE_6 | | | | | | | | DATA_BYTE_7 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_PL1_HI field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-24**<br>**DATA_BYTE_4** | RW | Payload Filter 1 high order bits for Pretended Networking payload filtering corresponding to data byte 4. |
| **23-16**<br>**DATA_BYTE_5** | RW | Payload Filter 1 high order bits for Pretended Networking payload filtering corresponding to data byte 5. |
| **15-8**<br>**DATA_BYTE_6** | RW | Payload Filter 1 high order bits for Pretended Networking payload filtering corresponding to data byte 6. |
| **7-0**<br>**DATA_BYTE_7** | RW | Payload Filter 1 high order bits for Pretended Networking payload filtering corresponding to data byte 7. |

### 60.8.34 FLEXCAN_FLT_ID2_IDMASK (Pretended Networking ID Filter 2 Register/ ID Mask Register)

Offset : B1Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IDE_MSK | RTR_MSK | FLT_ID2_IDMASK | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FLT_ID2_IDMASK | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_FLT_ID2_IDMASK field descriptions**

| Field | Type | Description |
|---|---|---|
| **30**<br>**IDE_MSK** | RW | ID Extended Mask Bit<br>This bit indicates whether the frame format (standard/extended) is used as part of the ID reception filter.<br>0b - The corresponding bit in the filter is "don't care"<br>1b - The corresponding bit in the filter is checked |
| **29**<br>**RTR_MSK** | RW | Remote Transmission Request Mask Bit<br>This bit indicates whether the frame type (data/remote) is part of the ID reception filter.<br>0b - The corresponding bit in the filter is "don't care"<br>1b - The corresponding bit in the filter is checked |

| | | |
|---|---|---|
| **28-0**<br>**FLT_ID2_IDMASK** | RW | ID Filter 2 for Pretended Networking Filtering / ID Mask Bits for Pretended Networking ID Filtering<br>This register is used to define either the ID filter value in extended frame format (29 bits), considering the FLT_ID2[28:0], or the ID filter value in standard frame format (11 bits), considering the FLT_ID2[28:18] (other bits in the [17:0] range have no meaning). Used only in range of ID filtering.<br>It can also be used to define either the mask values for the extended frame format (29 bits), considering the IDMASK[28:0], or for the standard frame format (11 bits), considering the IDMASK[28:18] (other bits in the [17:0] range have no meaning). Used only in exact ID filtering. |

### 60.8.35 FLEXCAN_PL2_PLMASK_LO (Pretended Networking Payload Low Filter 2 Register/ Payload Low Mask Register)

Offset : B20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA_BYTE_0 | | | | | | | | DATA_BYTE_1 | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA_BYTE_2 | | | | | | | | DATA_BYTE_3 | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_PL2_PLMASK_LO field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**DATA_BYTE_0** | RW | Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to the data byte 0. |
| **23-16**<br>**DATA_BYTE_1** | RW | Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to the data byte 1. |
| **15-8**<br>**DATA_BYTE_2** | RW | Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to the data byte 2. |

| | | |
|---|---|---|
| **7-0**<br>**DATA_BYTE_3** | RW | Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to the data byte 3. |

### 60.8.36 FLEXCAN_PL2_PLMASK_HI (Pretended Networking Payload High Filter 2 Low Order Bits/ Payload High Mask Register)

Offset : B24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA_BYTE_4 | | | | | | | | DATA_BYTE_5 | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA_BYTE_6 | | | | | | | | DATA_BYTE_7 | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_PL2_PLMASK_HI field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**DATA_BYTE_4** | RW | Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to the data byte 4. |
| **23-16**<br>**DATA_BYTE_5** | RW | Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to the data byte 5. |
| **15-8**<br>**DATA_BYTE_6** | RW | Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to the data byte 6. |
| **7-0**<br>**DATA_BYTE_7** | RW | Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to the data byte 7. |

### 60.8.37 FLEXCAN_WMB_CSn (Wake Up Message Buffer Register for C/S)

Offset : B40h+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | SRR | IDE | RTR | DLC | | | |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_WMB_CSn field descriptions**

| Field | Type | Description |
|---|---|---|
| **22**<br>**SRR** | RO | Substitute Remote Request<br>This bit can be received either recessive or dominant. |
| **21**<br>**IDE** | RO | ID Extended Bit<br>This bit identifies whether the frame format is standard or extended.<br>0b - Frame format is standard<br>1b - Frame format is extended |
| **20**<br>**RTR** | RO | Remote Transmission Request Bit<br>This bit identifies whether the frame is remote or not.<br>0b - Frame is data one (not remote)<br>1b - Frame is a remote one |
| **19-16**<br>**DLC** | RO | Length of Data in Bytes<br>This 4-bit field is the length (in bytes) of the Rx data received when FLEXCAN is in Pretended Networking mode. This 4-bit field is written by the FLEXCAN module, copied from the DLC (Data Length Code) field of the received frame. The DLC field indicates which data bytes are valid. |

### 60.8.38 FLEXCAN_WMB_IDn (Wake Up Message Buffer Register for ID)

Offset : B44h+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | ID | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ID | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_WMB_IDn field descriptions**

| Field | Type | Description |
|---|---|---|
| **28-0**<br>**ID** | RO | Received ID under Pretended Networking mode<br>This register stores either the 29 bits of the extended frame format (considering the ID[28:0] field), or the 11 bits of the standard frame format (considering the ID[28:18] field only; the remaining bits in the ID[17:0] range have no meaning). |

### 60.8.39 FLEXCAN_WMB_D03_n (Wake Up Message Buffer Register for Data 0-3)

Offset : B48h+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA_BYTE_0 | | | | | | | | DATA_BYTE_1 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA_BYTE_2 | | | | | | | | DATA_BYTE_3 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_WMB_D03_n field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**DATA_BYTE_0** | RO | Received payload corresponding to the data byte 0 under Pretended Networking mode |
| **23-16**<br>**DATA_BYTE_1** | RO | Received payload corresponding to the data byte 1 under Pretended Networking mode |
| **15-8**<br>**DATA_BYTE_2** | RO | Received payload corresponding to the data byte 2 under Pretended Networking mode |
| **7-0**<br>**DATA_BYTE_3** | RO | Received payload corresponding to the data byte 3 under Pretended Networking mode |

### 60.8.40 FLEXCAN_WMB_D47_n (Wake Up Message Buffer Register Data 4-7)

Offset : B4Ch+N*10h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA_BYTE_4 | | | | | | | | DATA_BYTE_5 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA_BYTE_6 | | | | | | | | DATA_BYTE_7 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_WMB_D47_n field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**DATA_BYTE_4** | RO | Received payload corresponding to the data byte 4 under Pretended Networking mode |
| **23-16**<br>**DATA_BYTE_5** | RO | Received payload corresponding to the data byte 5 under Pretended Networking mode |
| **15-8**<br>**DATA_BYTE_6** | RO | Received payload corresponding to the data byte 6 under Pretended Networking mode |
| **7-0**<br>**DATA_BYTE_7** | RO | Received payload corresponding to the data byte 7 under Pretended Networking mode |

### 60.8.41 FLEXCAN_EPRS (Enhanced CAN Bit Timing Prescalers Register)

Offset : BF0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | EDPRESDIV | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | ENPRESDIV | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_EPRS field descriptions**

| Field | Type | Description |
|---|---|---|
| **25-16**<br>**EDPRESDIV** | RW | Extended Data Phase Prescaler Division Factor<br>This field defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency in the data phase of a CAN FD message when CTRL2[BTE] is set.<br>The Sclock period defines the time quantum of the CAN FD protocol for the data bit rate. Sclock frequency = PE clock frequency / (EDPRESDIV + 1).<br>NOTE: To minimize errors when processing FD frames, use the same value for ENPRESDIV and EDPRESDIV. For more details see the first NOTE in section CAN FD frames. |
| **9-0**<br>**ENPRESDIV** | RW | Extended Nominal Prescaler Division Factor<br>This field defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency when CTRL2[BTE] is set; otherwise it reads as zero and a write operation has no effect.<br>The Sclock period defines the time quantum of the CAN protocol in the nominal phase. For the reset value, the Sclock frequency is equal to the PE clock frequency (see Protocol timing).<br>Sclock frequency = PE clock frequency / (ENPRESDIV + 1) |

### 60.8.42 FLEXCAN_ENCBT (Enhanced Nominal CAN Bit Timing Register)

Offset : BF4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | NRJW | | | | | | | 0 | | | NTSEG2 | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NTSEG2 | | | 0 | | | | | NTSEG1 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_ENCBT field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **28-22**<br>**NRJW** | RW | Nominal Resynchronization Jump Width<br>This field defines the maximum number of time quanta that a nominal bit time can be changed by one resynchronization when CTRL2[BTE] bit is asserted; otherwise it has no effect.<br>One time quantum is equal to the Sclock period. Nominal Resync Jump Width = NRJW + 1. |
| **18-12**<br>**NTSEG2** | RW | Nominal Time Segment 2<br>This field defines the length of Time Segment 2 in the nominal bit time when CTRL2[BTE] is set; otherwise it has no effect.<br>Nominal Time Segment 2 = (NTSEG2 + 1) × Time-Quanta. Time-Quantum = one Sclock period. |
| **7-0**<br>**NTSEG1** | RW | Nominal Time Segment 1<br>This field defines the length of Time Segment 1 in the bit time when CTRL2[BTE] is set; otherwise it has no effect.<br>Nominal Time Segment 1 = (NTSEG1 + 1) × Time-Quanta. Time-Quantum = one Sclock period. |

### 60.8.43 FLEXCAN_EDCBT (Enhanced Data Phase CAN bit Timing Register)

Offset : BF8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | DRJW | | | | 0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DTSEG2 | | | | 0 | | | | | | DTSEG1 | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_EDCBT field descriptions**

| Field | Type | Description |
|---|---|---|
| **25-22**<br>**DRJW** | RW | Data Phase Resynchronization Jump Width<br>This field defines the maximum number of time quanta that a data phase bit time can be changed by one resynchronization when CTRL2[BTE] is set; otherwise it has no effect.<br>Data Phase Resync Jump Width = DRJW + 1. |

| | | |
|---|---|---|
| **15-12**<br>**DTSEG2** | RW | Data Phase Time Segment 2<br>This field defines the length of time segment 2 in the data phase bit time when CTRL2[BTE] is set; otherwise it has no effect.<br>Data Phase Time Segment 2 = (DTSEG2 + 1) x Time-Quanta. Time-Quantum = one Sclock period. |
| **4-0**<br>**DTSEG1** | RW | Data Phase Segment 1<br>This field defines the length of time segment 1 in the data phase bit time when CTRL2[BTE] is set; otherwise it has no effect.<br>Data Phase Time Segment 1 = (DTSEG1 + 1) x Time-Quanta.<br>Time-Quantum = one Sclock period. |

### 60.8.44 FLEXCAN_ETDC (Enhanced Transceiver Delay Compensation Register)

Offset : BFCh

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ETDCEN | TDMDIS | 0 | | | | | | | ETDCOFF | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ETDCFAIL | 0 | | | | | | | ETDCVAL | | | | | | | |
| W | w1c | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_ETDC field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**ETDCEN** | RW | Transceiver Delay Compensation Enable<br>This bit can be used for enabling and disabling the TDC feature. It can be written in Freeze mode only.<br>NOTE: Refer to Transmitter delay compensation in the CAN Protocol standard (ISO 11898-1) for details.<br>NOTE: TDC must be disabled when the Loop Back Mode is enabled (see CTRL1[LPB] register).<br>0b - TDC is disabled<br>1b - TDC is enabled |

| | | |
|---|---|---|
| **30**<br>**TDMDIS** | RW | Transceiver Delay Measurement Disable<br>This bit can be used for disabling the transceiver delay measurement. When the TDC measurement is disabled, the secondary sample point position is determined only by the enhanced TDC offset (ETDCOFF) field. Otherwise, if the TCD measurement is enabled, then the secondary sample point position is determined by the sum of the transceiver delay measurement plus the enhanced TDC offset.<br>This field is not affected by soft reset.<br>NOTE: This bit can be enabled only if CTRL2[BTE] is set.<br>0b - TDC measurement is enabled<br>1b - TDC measurement is disabled |
| **22-16**<br>**ETDCOFF** | RW | Enhanced Transceiver Delay Compensation Offset<br>This field contains the offset value to be added to the measured transceiver's loop delay in order to define the position of the delayed comparison point when bit rate switching is active. See Transceiver delay compensation for more details on how the loop delay measurement is performed.<br>ETDCOFF can be written in Freeze mode only. Its value can be defined in protocol engine (PE) clock periods (CANCLK, see Protocol timing for more details), and must be selected to be smaller than the CAN bit duration in the data bit rate for proper operation.<br>NOTE: ETDCOFF must not be configured as zero.<br>NOTE: If CTRL2[BTE] is set after a chip level hard reset, ETCDOFF is read as 0x1. |
| **15**<br>**ETDCFAIL** | W1C | Transceiver Delay Compensation Fail<br>This bit indicates when the Transceiver Delay Compensation (TDC) mechanism is out of range, unable to compensate the transceiver's loop delay and successfully compare the delayed received bits to the transmitted ones (see Transceiver delay compensation . ETDCFAIL sets in the first time FLEXCAN detects the out of range condition. The CPU needs to write 1 to clear it.<br>0b - Measured loop delay is in range.<br>1b - Measured loop delay is out of range. |
| **7-0**<br>**ETDCVAL** | RO | Enhanced Transceiver Delay Compensation Value<br>This register contains the ETDCOFF field added to the measured value of the transceiver loop delay in the latest transmitted CAN FD frame, with BRS equal to recessive.<br>This field is only updated by the hardware if ETDC[ETDCEN] is set. This field is affected by soft reset.<br>NOTE: If TDMDIS is set, ETDCVAL stores ETDCOFF only. |

### 60.8.45 FLEXCAN_FDCTRL (CAN FD Control Register)

Offset : C00h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FDRATE | 0 | | | | | | | | | | | | | MBDSR0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TDCEN | TDCFAIL | 0 | TDCOFF | | | | | 0 | | TDCVAL | | | | | |
| W | | w1c | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_FDCTRL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**FDRATE** | RW | Bit Rate Switch Enable<br>This bit enables the effect of the Bit Rate Switch (BRS bit) during the data phase of Tx messages.<br>The CPU can write this bit any time. However, its effect turns active only when the FLEXCAN bus is in Wait for Bus Idle, Bus Idle, or Bus Off state, or when the current frame under reception or transmission reaches the interframe space.<br>By negating FDCTRL[FDRATE], the CPU can force all bits in CAN FD messages to be transmitted in nominal bit rate, no matter what the value is in the BRS bit of the Tx MBs.<br>0b - Transmit a frame in nominal rate. The BRS bit in the Tx MB has no effect.<br>1b - Transmit a frame with bit rate switching if the BRS bit in the Tx MB is recessive. |
| **17-16**<br>**MBDSR0** | RW | Message Buffer Data Size for Region 0<br>This two-bit field selects the data size (8, 16, 32, or 64 bytes) for region R0 of message buffers allocated in RAM.<br>It can be written in Freeze mode only.<br>00b - Selects 8 bytes per message buffer.<br>01b - Selects 16 bytes per message buffer.<br>10b - Selects 32 bytes per message buffer.<br>11b - Selects 64 bytes per message buffer. |

| | | |
|---|---|---|
| **15**<br>**TDCEN** | RW | Transceiver Delay Compensation Enable<br>This bit can be used to enable and disable the TDC feature. It can be written in Freeze mode only.<br>NOTE: See Transmitter delay compensation in the CAN Protocol standard (ISO 11898-1) for details.<br>NOTE: TDC must be disabled when Loop Back mode is enabled (see CTRL1[LPB]).<br>NOTE: If CTRL2[BTE] is set, TDCEN is read as zero and a write operation has no effect.<br>0b - TDC is disabled<br>1b - TDC is enabled |
| **14**<br>**TDCFAIL** | W1C | Transceiver Delay Compensation Fail<br>This bit indicates when the Transceiver Delay Compensation (TDC) mechanism is out of range, unable to compensate the transceiver's loop delay and successfully compare the delayed received bits to the transmitted ones (see Transceiver delay compensation). TDCFAIL sets in the first time FLEXCAN detects the out of range condition. The CPU needs to write one to clear it.<br>NOTE: If CTRL2[BTE] is set, TDCFAIL is read as zero and a write operation has no effect.<br>0b - Measured loop delay is in range.<br>1b - Measured loop delay is out of range. |
| **12-8**<br>**TDCOFF** | RW | Transceiver Delay Compensation Offset<br>This bit field contains the offset value to be added to the measured transceiver's loop delay in order to define the position of the delayed comparison point when bit rate switching is active. See Transceiver delay compensation for more details on how the loop delay measurement is performed.<br>TDCOFF can be written in Freeze mode only. Its value can be defined in Protocol Engine (PE) Clock periods (CANCLK, see Protocol timing for more details), and must be selected to be smaller than the CAN bit duration in the data bit rate for proper operation.<br>NOTE: If CTRL2[BTE] is set, TDCOFF is read as zero and a write operation has no effect.<br>NOTE: It is not recommended to have TDCOFF equal to zero. |

| | | |
|---|---|---|
| **5-0**<br>**TDCVAL** | RO | Transceiver Delay Compensation Value<br>This register contains the value of the transceiver loop delay measured from the transmitted EDL to R0 transition edge to the respective received one added to the TDCOFF value specified in the FDCTRL register. This value is an integer multiple of the Protocol Engine (PE) Clock period (CANCLK).<br>NOTE: If CTRL2[BTE] is set, TDCVAL is read as zero.<br>See Protocol timing for more details on how the loop delay measurement is performed. |

### 60.8.46 FLEXCAN_FDCBT (CAN FD Bit Timing Register)

Offset : C04h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | FPRESDIV | | | | | | | | | | 0 | FRJW | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | FPROPSEG | | | | | 0 | FPSEG1 | | | | 0 | FPSEG2 | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_FDCBT field descriptions**

| Field | Type | Description |
|---|---|---|
| **29-20**<br>**FPRESDIV** | RW | Fast Prescaler Division Factor<br>This 10-bit field defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency in the data bit rate portion of a CAN FD message with the BRS bit set.<br>The Sclock period defines the time quantum of the CAN FD protocol for the data bit rate. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Sclock frequency = PE clock frequency / (FPRESDIV + 1).<br>NOTE: To minimize errors when processing FD frames, use the same value for FPRESDIV and PRESDIV (in CBT or CTRL1). For more details see the first NOTE in section CAN FD frames. |

| | | |
|---|---|---|
| **18-16**<br>**FRJW** | RW | Fast Resync Jump Width<br>This 3-bit field defines the maximum number of time quanta that a bit time can be changed by one resynchronization in the data bit rate portion of a CAN FD message with the BRS bit set.<br>One time quantum is equal to the Sclock period. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Resync Jump Width = FSJW + 1. |
| **14-10**<br>**FPROPSEG** | RW | Fast Propagation Segment<br>This 5-bit field defines the length of the propagation segment in the bit time in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Propagation Segment Time = FPROPSEG × Time-Quanta.<br>Time-Quantum = one Sclock period. |
| **7-5**<br>**FPSEG1** | RW | Fast Phase Segment 1<br>This 3-bit field defines the length of phase segment 1 in the bit time in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Phase Segment 1 = (FPSEG1 + 1) × Time-Quanta.<br>Time-Quantum = one Sclock period. |
| **2-0**<br>**FPSEG2** | RW | Fast Phase Segment 2<br>This 3-bit field defines the length of phase segment 2 in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in Freeze mode because it is blocked by hardware in other modes.<br>Phase Segment 2 = (FPSEG2 + 1) × Time-Quanta.<br>Time-Quantum = one Sclock period. |

### 60.8.47 FLEXCAN_FDCRC (CAN FD CRC Register)

Offset : C08h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | FD_MBCRC | | | | | | | 0 | | | FD_TXCRC | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FD_TXCRC | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_FDCRC field descriptions**

| Field | Type | Description |
|---|---|---|
| **30-24**<br>**FD_MBCRC** | RO | CRC Mailbox Number for FD_TXCRC<br>This field indicates the number of the mailbox corresponding to the value in the FD_TXCRC field, for both FD and non-FD frames.<br>It reports the same information as in CRCR[MBCRC]. |
| **20-0**<br>**FD_TXCRC** | RO | Extended Transmitted CRC value<br>This 21-bit field contains the CRC value calculated over the most recent transmitted message. Different CRC polynomials are used for different frame formats. A 15-bit polynomial, CRC_15, is used for all frames in CAN format. The second 17-bit polynomial, CRC_17, is used for frames in CAN FD format with a data field up to sixteen bytes long. The third 21-bit polynomial, CRC_21, is used for frames in CAN FD format with a data field longer than sixteen bytes.<br>For CRC_15 and CRC_17, the 6 most significant bits and the 4 most significant bits are reported as zeros, respectively.<br>For CRC_15, this register has the same content as CRC register. |

### 60.8.48 FLEXCAN_ERFCR (Enhanced Rx FIFO Control Register)

Offset : C0Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ERFEN | DMALW | | | | | 0 | | NEXIF | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | NFE | | | | | | | 0 | ERFWM | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_ERFCR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31**<br>**ERFEN** | RW | Enhanced Rx FIFO enable<br>This field enables the Enhanced Rx FIFO.<br>NOTE: ERFEN must not be set if MCR[RFEN] is set.<br>0b - Enhanced Rx FIFO is disabled<br>1b - Enhanced Rx FIFO is enabled |
| **30-26**<br>**DMALW** | RW | DMA Last Word<br>This field defines the last DMA address for each Enhanced RX FIFO element.<br>0-last FIFO address:0x2000<br>1-last FIFO address:0x2004<br>2-last FIFO address:0x2008<br>…<br>19-last FIFO address:0x204c<br>Note: undefined DMALW values are reserved and must not be used |
| **22-16**<br>**NEXIF** | RW | Number of Extended ID Filter Elements<br>This field defines the number of extended ID filter elements used during the Enhanced Rx FIFO matching process.NEXIF must satisfy this condition:<br>NEXIF <= NFE + 1<br>The number of standard ID filter elements can be determined by this equation.<br>Number of Standard ID Filter Elements = 2 x (NFE - NEXIF +1) |
| **13-8**<br>**NFE** | RW | Number of Enhanced Rx FIFO Filter Elements<br>This field defines the total number of filter elements used during the enhanced RX FIFO matching process |
| **4-0**<br>**ERFWM** | RW | Enhanced Rx FIFO Watermark<br>This field defines the minimum number of CAN messages stored in the Enhanced RX FIFO. When that number is reached, then ERFSR[ERFWMI] is set.<br>Minimum number of CAN messages = ERFWM + 1. |

### 60.8.49 FLEXCAN_ERFIER (Enhanced Rx FIFO Interrupt Enable Register)

Offset : C10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ERFUFWIE | ERFOVFIE | ERFWMIIE | ERFDAIE | 0 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

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

**FLEXCAN_ERFIER field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**ERFUFWIE** | RW | Enhanced Rx FIFO Underflow Interrupt Enable This field enables interrupt for ERFSR[ERFUFW].<br>0b - Enhanced Rx FIFO Underflow interrupt is disabled<br>1b - Enhanced Rx FIFO Underflow interrupt is enabled |
| **30**<br>**ERFOVFIE** | RW | Enhanced Rx FIFO Overflow Interrupt Enable This field enables interrupt for ERFSR[ERFOVF].<br>0b - Enhanced Rx FIFO Overflow is disabled<br>1b - Enhanced Rx FIFO Overflow is enabled |
| **29**<br>**ERFWMIIE** | RW | Enhanced Rx FIFO Watermark Indication Interrupt Enable This field enables interrupt for ERFSR[ERFWMI].<br>0b - Enhanced Rx FIFO Watermark interrupt is disabled<br>1b - Enhanced Rx FIFO Watermark interrupt is enabled |
| **28**<br>**ERFDAIE** | RW | Enhanced Rx FIFO Data Available Interrupt Enable This field enables interrupt for ERFSR[ERFDA].<br>0b - Enhanced Rx FIFO Data Available interrupt is disabled<br>1b - Enhanced Rx FIFO Data Available interrupt is enabled |

### 60.8.50 FLEXCAN_ERFSR (Enhanced Rx FIFO Status Register)

Offset : C14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ERFUFW | ERFOVF | ERFWMI | ERFDA | ERFCLR | 0 | | | | | | | | | ERFE | ERFF |
| W | w1c | w1c | w1c | w1c | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | ERFEL | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_ERFSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**ERFUFW** | W1C | Enhanced Rx FIFO Underflow<br>This field indicates that an underflow condition occurred in the enhanced Rx FIFO. This field generates an interrupt if ERFIER[ERFUFWIE] is set.<br>0b - No such occurrence<br>1b - Enhanced Rx FIFO underflow |
| **30**<br>**ERFOVF** | W1C | Enhanced Rx FIFO Overflow<br>This field indicates that an overflow condition occurred in the Enhanced Rx FIFO. This field generates an interrupt if ERFIER[ERFOVFIE] is set.<br>0b - No such occurrence<br>1b - Enhanced Rx FIFO overflow |
| **29**<br>**ERFWMI** | W1C | Enhanced Rx FIFO Watermark Indication<br>This field is set by the hardware if the number of messages available in the Enhanced Rx FIFO is greater than the watermark defined in ERFCR[ERFWM].<br>This field generates an interrupt if ERFIER[ERFWMIIE] is set.<br>0b - No such occurrence<br>1b - The number of messages in FIFO is greater than the watermark |
| **28**<br>**ERFDA** | W1C | Enhanced Rx FIFO Data Available<br>This field is set by the hardware when there is at least one message stored in the ERX FIFO. This field generates an interrupt if ERFIER[ERFDAIE] is set.<br>0b - No such occurrence<br>1b - There is at least one message stored in Enhanced Rx FIFO |
| **27**<br>**ERFCLR** | RW | Enhanced Rx FIFO Clear<br>Writing one to this field during Freeze mode clears Enhanced Rx FIFO content. |
| **17**<br>**ERFE** | RO | Enhanced Rx FIFO empty<br>This field indicates that Enhanced Rx FIFO is empty.<br>0b - Enhanced Rx FIFO is not empty<br>1b - Enhanced Rx FIFO is empty |
| **16**<br>**ERFF** | RO | Enhanced Rx FIFO full<br>This field indicates that Enhanced Rx FIFO is full.<br>0b - Enhanced Rx FIFO is not full<br>1b - Enhanced Rx FIFO is full |
| **5-0**<br>**ERFEL** | RO | Enhanced Rx FIFO Elements<br>This field indicates the number of CAN messages stored in the Enhanced Rx FIFO. |

### 60.8.51 FLEXCAN_GATE_CS (Gateway Control Register)

Offset : 1A00h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | GATE_SEP | | | | | | | | 0 | | | | | | | GATE_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_GATE_CS field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-8 GATE_SEP** | RW | Gate separate, higher than value in the field would use top header |
| **0 GATE_EN** | RW | Gate enable |

### 60.8.52 FLEXCAN_GATE_BUFFERn (Gateway Buffer Register)

Offset : 1A10h+N*4h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NEW_HEADER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NEW_HEADER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_GATE_BUFFERn field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-0**<br>**NEW_HEADER** | RW | New header, allocated in RAM<br>Word 0: Frame header 0<br>Word 1: New ID0<br>Word 2: Frame header 1<br>Word 3: New ID1 |

### 60.8.53 FLEXCAN_GATE_DUMMYn (Gateway Router Dummy Register)

Offset : 1A80h+N*4h (N : 0 ~ 17)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GATE_DUMMY | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GATE_DUMMY | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_GATE_DUMMYn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**GATE_DUMMY** | RW | Dummy address for message gate |

### 60.8.54 FLEXCAN_ERX_FIFOn (Enhanced Rx FIFO Register)

Offset : 2000h+N*4h (N : 0 ~ 239)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_ERX_FIFOn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**RX** | RW | Enhanced Rx FIFO<br>This bit receives CAN messages when ERFCR[ERFEN] is asserted. |

### 60.8.55 FLEXCAN_ERFFELn (Enhanced Rx FIFO Filter Element Register)

Offset : 3000h+N*4h (N : 0 ~ 31)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FEL | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FEL | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**FLEXCAN_ERFFELn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**FEL** | RW | Filter Element Bits<br>Each filter element is used during the match process and, if the matching criteria are met, a message is stored in the Enhanced Rx FIFO. |

# Chapter 61 Micro Second Channel (MSC)

## 61.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • Added the description about enabling the timeout interrupt in Section *Frame Watchdog;*<br>• Updated the description of the "WP_EN" bit field in the register section *MSC_GCR;*<br>• Updated the descriptions of the bit fields "DTS" and "CTS" in the register section *MSC_TCCTR;*<br>• Updated the bit field descriptions in the register section *MSC_ISCR;*<br>• Updated the description of the "RDONE" bit field in the register section *MSC_MSR;*<br>• Editorial changes |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 61.2 Chip-specific Information

There is one Micro Second Channel (MSC) interface instance for the FC7240F2M. The acceptable maximum frequency of the MSC functional clock is 100 MHz.

## 61.3 Features

The feature set includes:

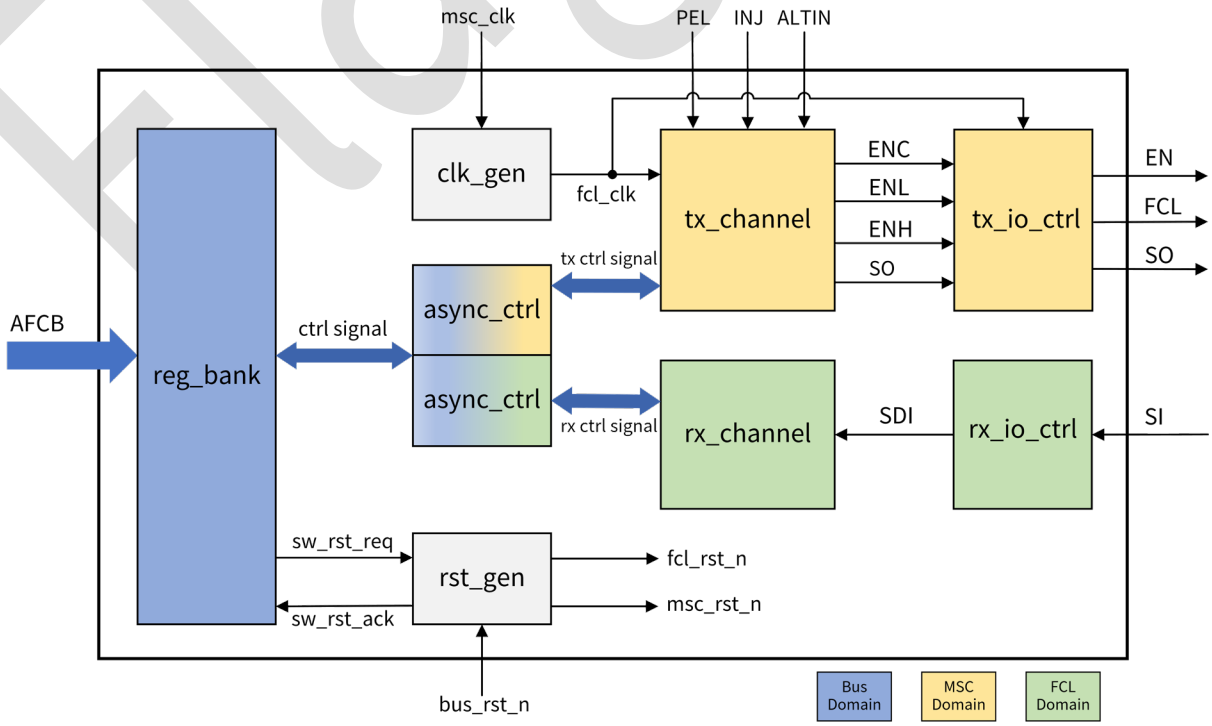
- Synchronous transmitting channel
  - Baud rate: $f_{FCL} = f_{MSC}/2$
  - Three types of frames: command, data, and passive frame
  - Optional SEL bit in command and data frames
  - Emergency stop without CPU intervention
  - Trigger and repetition modes are supported
- Asynchronous receiving channel
  - Baud rate: $f_{MSC}$ divided by 4, 8, 16, 32, 64, 128, or 256
  - Error diagnosis including parity error and stop bit error
  - Spiker filter for asynchronous serial reception
  - 4-out-of-1 channel selector
  - Optional address field in each frame

## 61.4 Overview

The MSC interface is designed to have communication typically between the SoC and power switches or other peripheral devices. It includes a high-speed synchronous transmitting channel and a low-speed asynchronous receiving channel. The former is compatible with the SPI and the latter can receive UART data. Up to four slaves with this interface can be linked with a dynamic chip selection switch.

A simplified block diagram of the MSC interface module is shown below.

**Figure 140. MSC block diagram**



Sub-module reg_bank implements registers write/read logic in the bus clock domain. Other modules are in the function clock domain. For the transmitting channel, tx_channel is the core control module whose FSMs are used to have control of frame scheduling. And tx_io_ctrl is responsible for polarity control and channel selection operation. For the receiving channel, the filter input SI and one stream are selected in rx_io_ctrl. And then rx_channel samples data and checks the validation of each frame received. The signal ALTIN[31:0] is used as the data of the tx_channel data frame. The signal INJ[1:0] is used to change the tx_channel by inserting a bit. Emergency stop signals are used to control the priority of the data frame.

## 61.5 Functional Description

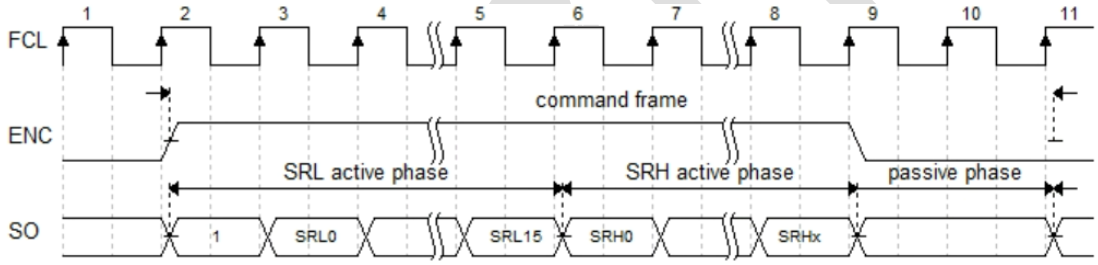
In this section, the function of the MSC is described in detail, including the transmitting channel and the receiving channel.

### 61.5.1 Transmitting Channel

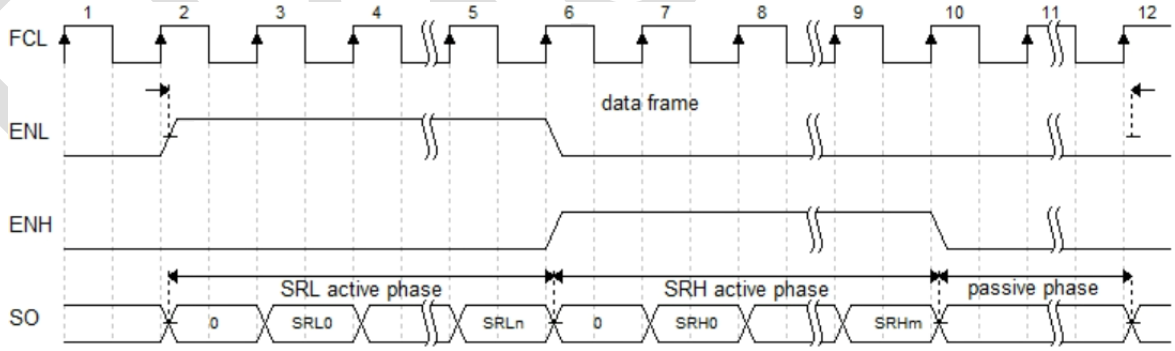
#### *61.5.1.1 Frame Definition*

A 32-bit shift register is in the transmitting channel logic. It is divided into two 16-bit parts, the shift register low part (SRL) and shift register high part (SRH). The content inside the transmitting channel data register TCDAR, inside the transmitting channel command register TCCOR, or inputs on the alternate input (ALTIN) can be selected as the data source of each bit.

For a command frame, the high-level selection bit must be existing no matter what the value of TCCTR.SELL. So, the command frame is indicated by it. The number of bits transmitted during SRL and SRH is defined by TCCTR.NBS. And the length of the passive phase is fixed as two FCL cycles.




For a data frame, the low-level selection bit exists when enabling TCCTR.SELL or TCCTR.SELH. TCCTR.NHBS determines the length of the SRH active phase and TCCTR.NLBS determines the length of the SRL active phase. The duration of the data frame passive phase is determined by TCCTR.PL.




A passive time frame is generated only in repetition mode. The data output is considered invalid. ENL and ENH remain at a low level during this kind of frame. FCL may not toggle according to IOCR.FCLCTRL.

***Note:***

- ENL indicates the SRL active phase of a data frame.
- ENH indicates the SRH active phase of a data frame.
- ENC indicates the active phase of a command frame.
- SO indicates the serial data output.

#### 61.5.1.2 Source Selection

The transmitting channel command register TCCOR is the source of a frame when TCCTR.CTS is high. Otherwise, a 1-out-of-3 selector is used to choose the transmitting channel data register TCDAR, ALTIN or bitwise inversion of ALTIN as the data source of each bit. When TCDAR is selected, optional injection logic helps to tie two bits of TCDAR with specified bit values on the SoC level if this feature is enabled. But TCDAR will be selected in case of emergency stop enabled no matter what value of transmitting channel source selection low/high register TCSLR/TCSHR.

#### 61.5.1.3 Transmission Mode

In the transmitting channel, two modes work: trigger mode and repetition mode. Program TCCTR.WM to choose one mode before frame transmission.

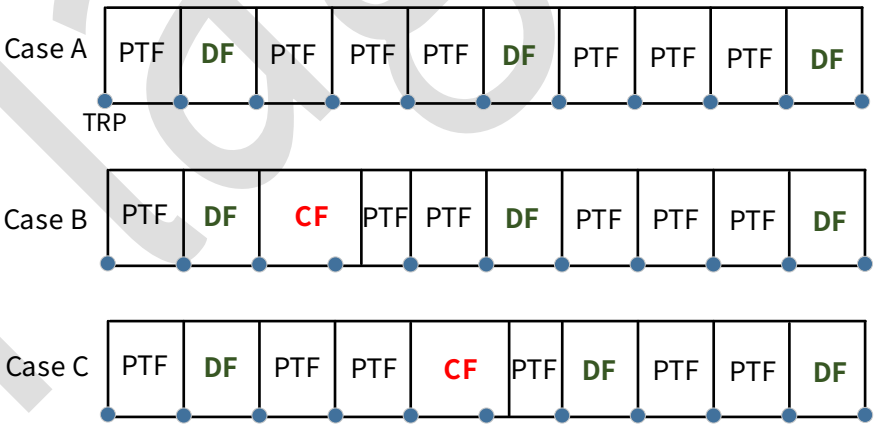
In trigger mode, command frame or data frame transmissions start when TCCTR.CTS and TCCTR.DTS is set respectively. Software writing TCCOR can launch one command frame while sets ISCR.SDTS, then one data frame can send out. There is a queuing mechanism that frame transmission is delayed and started when the active one has been finished if a frame is currently output. Please note that the command frame always has priority over the data frame, which means a command frame will be launched once TCCTR.CTS is set.

In repetition mode, data frames are sent out continuously at a specific interval without any CPU intervention. Passive time frames will be inserted automatically in the time gap between two data frames. The application also can insert command frames by setting ISCR.SCTS. The rules about frame transmission are as follows in this mode.

1) Data frame (DF) or command frame (CF) transmission only starts at time reference points (TRP)
2) Passive time frame (PTF) number between two DF is defined by TCSTR.NP if no command frame insertion
3) Active frames (DF or CF) cannot occur consecutively, there must be at least one PTF in between
4) A CF needs to be sent or is being sent at a time when a DF should have been sent, this DF is delayed

A source selection event occurs just before every TRP so that data is updated in time.

**Figure 141. Data repetition mode frame example**



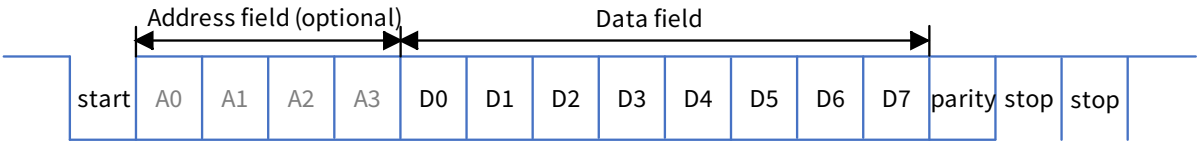

In this way, the time interval between two DFs must be less than the specified value. $t_{PTF}$ is equal to $t_{DF}$ when no CF insertion. But CF will continue to the next TRP if a CF inserts before it (case C). Please note that we assume there is a select bit for SRL when calculating $t_{DF}$, which means TCCTR.SELL does not influence the time between two consecutive TRPs.

#### 61.5.1.4 Baud Rate

The baud rate value is fixed to $f_{MSC}/2$. And $f_{MSC}$ is dependent on the SoC peripheral clock configuration.

### 61.5.2 Receiving Channel

#### *61.5.2.1 Frame Definition*




There are two types of frames in the receiving channel: 12-bit frame and 16-bit frame with address field. The frame consists of the following parts:

- One low-level start bit
- 8-bit data field
- Optional 4-bit address field
- One parity bit and two high-level stop bits

#### *61.5.2.2 Parity Checking and Error Reporting*

Bit RCCSR.PCTL implements parity control, even or odd. With RCCSR.PCTL=0, even parity mode is selected. Odd parity mode is enabled with RCCSR.PCTL is set.

The receiver compares the parity value received with the calculated one. Error report if the two are not equal. And stop bit field also be detected by it. For example, only one stop bit or no stop bit will also trigger an error report. The user can read reception data register RDRx.RERR to figure out what happened.

#### *61.5.2.3 Data Decode*

When a start bit is received, reception is enabled and begins to sample the data stream and buffer it into a shift register. The operation will be finished until enough stop bits have been detected. After that, the content of the shift register is loaded into RDRx, and RDRx.VLD is set to indicate a frame has been received, which is independent if RDRx.RERR is zero or not.

For a 12-bit frame, the payload and the corresponding information of it are stored in RDR0 register. While for a 16-bit frame, the two most significant address bits ADDR[3:2] of the address field select which RDRx as the target buffer. Other bits ADDR[1:0] will be stored in RDRx.LAF.

#### *61.5.2.4 Baud Rate*

The baud rate of the receiving channel is derived from the function clock $f_{MSC}$. A divider is used to output a sampling clock, and the frequency division value is determined by RCCSR.RBR. 4, 8, 6, 32, 64, 128, and 256 can be selected for that. Please note that the receiving channel is closed when RCCSR.RBR is programmed as 0 even if GCR.EN is set.

#### *61.5.2.5 Frame Watchdog*

A frame watchdog monitors the time interval between two frames' start bits. The timer reloads the starting value from the reception time-out register RTOR.TOV and starts counting downwards. An interrupt is generated when the timer is zero if enabled.

The counter reloads on events:

- Enable the receiving channel
- A frame start bit received with channel enabled

### 61.5.3 I/O Control

#### *61.5.3.1 Tx_io_ctrl*

ENC, ENL, and ENH are converted to EN [3:0] by output control logic. EN [x] will be asserted if any one of ENC, ENL, and ENH is active. And the user selects which bit of EN[x] is changed to the high level by programming input and output control register IOCR.ESC, ESL, and ESH respectively. Polarity selection using IOCR.ENP, IOCR.SOP and IOCR.FCLP. With IOCR.FCLCTRL = 1, the FCL will always be toggled independently whether a transmitting frame is transmitted or not. It is designed for some slaves that output data at a clock derived from FCL received from a master with an MSC interface.

#### *61.5.3.2 Rx_io_ctrl*

The MSC receiving channel can be connected to up to four slaves. Bit field IOCR.IDS selects one of them and then the field IOCR.IPS selects the polarity of the single-bit stream. The last step, before parsing the incoming data stream, a voter filters the noise. If two or more of the three continuous samplings are 1, it is determined as 1.

## 61.6 Application Information

The MSC is especially used to connect external power devices. Data or command information is sent out to the power device via a synchronous data stream. Data or status from a slave can be received via an asynchronous data stream. Up to four slaves can be attached to one MSC interface instance. External multi-channel PWM signals are passed into the MSC interface module to have independent control of multiple switches.

1) Control an external power switching unit.
2) Read status information from the power switching unit.
3) Connect to other peripheral devices.

## 61.7 Register Definition

MSC base address : 4047_4000h

**Register Overview - MSC**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **00h** | MSC_GCR (Global Control Register) | 32 | RW | 0000_0000h | page |
| **04h** | MSC_CCULR (Channel Control Unlock Register) | 32 | WO | 0000_0000h | page |
| **10h** | MSC_RCCSR (Reception Channel Control and Status Register) | 32 | RW | 0000_0000h | page |
| **14h** | MSC_TCCTR (Transmitting Channel Control Register) | 32 | RW | 0000_8000h | page |
| **18h** | MSC_TCSTR (Transmitting Channel Status Register) | 32 | RW | 0000_0000h | page |
| **1Ch** | MSC_TCDAR (Transmitting Channel Data Register) | 32 | RW | 0000_0000h | page |
| **20h** | MSC_TCCOR (Transmitting Channel Command Register) | 32 | RW | 0000_0000h | page |
| **24h** | MSC_TCSLR (Transmitting Channel Source Selection Low Register) | 32 | RW | 0000_0000h | page |
| **28h** | MSC_TCSHR (Transmitting Channel Source Selection High Register) | 32 | RW | 0000_0000h | page |
| **2Ch** | MSC_TCELR (Transmitting Channel Emergency Load Register) | 32 | RW | 0000_0000h | page |
| **30h** | MSC_RDR0 (Reception Data Register0) | 32 | RW | 0000_0000h | page |

| | | | | | |
|---|---|---|---|---|---|
| **34h** | MSC_RDR1 (Reception Data Register1) | 32 | RW | 0000_0000h | page |
| **38h** | MSC_RDR2 (Reception Data Register2) | 32 | RW | 0000_0000h | page |
| **3Ch** | MSC_RDR3 (Reception Data Register3) | 32 | RW | 0000_0000h | page |
| **40h** | MSC_INCR (Interrupt Control Register) | 32 | RW | 0000_0000h | page |
| **44h** | MSC_INSR (Interrupt Status Register) | 32 | RO | 0000_0000h | page |
| **48h** | MSC_ISCR (Interrupt Set/Clear Register) | 32 | WO | 0000_0000h | page |
| **4Ch** | MSC_IOCR (Input/Output Control Register) | 32 | RW | 0000_0000h | page |
| **58h** | MSC_TCCTR1 (Transmitting Channel Control Register1) | 32 | RW | 0000_0000h | page |
| **5Ch** | MSC_RTOR (Reception Timeout Register) | 32 | RW | 0000_0000h | page |
| **ECh** | MSC_SRCR (Software Reset Clear Register) | 32 | RW | 0000_0000h | page |
| **F4h** | MSC_MSR (Software Reset Register) | 32 | RW | 0000_0000h | page |

### 61.7.1 MSC_GCR (Global Control Register)

Offset : 00h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | WP_EN | EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_GCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **1**<br>**WP_EN** | RW | Write Protection Enable<br>Write protection enable control bit for channel control registers (TCCTR, RCCSR, and IOCR). This field can only be written 1 to enable the bit and the WP_EN can be cleared by writing 0x10248888 to CCULR or hardware reset and SW reset.<br>0b - Write protection for channel control registers is disabled.<br>1b - Write protection for channel control registers is enabled. |

| | | |
|---|---|---|
| **0**<br>**EN** | RW | Global Enable Control Bit<br>0b - Disable MSC<br>1b - Enable MSC |

### 61.7.2 MSC_CCULR (Channel Control Unlock Register)

Offset : 04h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | SEQ | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | SEQ | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_CCULR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**SEQ** | WO | Sequence Value<br>0x10248888 is needed while other values are not valid.<br>TCCTR, IOCR, RCCSR are unlocked when the unlock sequence is written into this field. |

### 61.7.3 MSC_RCCSR (Reception Channel Control and Status Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RX_BUSY | 0 | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | HIDC | PCTL | RBR | | | RFT |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_RCCSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**RX_BUSY** | RO | Receiving Channel Busy<br>0b - rx_channel is idle<br>1b - rx_channel is busy |
| **5**<br>**HIDC** | RW | Hardware Receive Interrupt Delay Control<br>0b - No delay<br>1b - Delay of 1-bit time |
| **4**<br>**PCTL** | RW | Parity Control<br>0b - Even parity mode is selected<br>1b - Odd parity mode is selected |
| **3-1**<br>**RBR** | RW | Receiving Channel Baud Rate<br>000b - Receiving channel is disabled<br>001b - Baud rate=fmsc/4<br>010b - Baud rate=fmsc/8<br>011b - Baud rate=fmsc/16<br>100b - Baud rate=fmsc/32<br>101b - Baud rate=fmsc/64<br>110b - Baud rate=fmsc/128<br>111b - Baud rate=fmsc/256 |
| **0**<br>**RFT** | RW | Receiving Channel Frame Type<br>0b - 12-bit frame selected<br>1b - 16-bit frame with 4-bit address field selected |

### 61.7.4 MSC_TCCTR (Transmitting Channel Control Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | PL | | | | 0 | | NBS | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TCDIS | SELH | SELL | NHBS | | | | | NLBS | | | | | DTS | CTS | WM |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_TCCTR field descriptions**

| Field | Type | Description |
|---|---|---|
| **28-24**<br>**PL** | RW | Passive Phase Length at Data Frames |
| **21-16**<br>**NBS** | RW | Number of Bits Shifted at Command Frames<br>The maximum acceptable value is 32 and programming it to more than 32 will result in unexpected behavior. |
| **15**<br>**TCDIS** | RO | Transmitting channel is disabled |
| **14**<br>**SELH** | RW | Selection bit for SRH at the beginning of a data frame. |
| **13**<br>**SELL** | RW | Selection bit for SRL at the beginning of a data frame. |
| **12-8**<br>**NHBS** | RW | Number of SRH Bits Shifted during a Data Frame<br>The maximum acceptable value is 16 and programming it to more than 16 will result in unexpected behavior. |
| **7-3**<br>**NLBS** | RW | Number of SRL Bits Shifted during a Data Frame<br>The maximum acceptable value is 16 and programming it more than 16 will result in unexpected behavior. |
| **2**<br>**DTS** | RO | Data Frames Need to Be Sent<br>If this bit is set, it indicates there are data frames wait to be sent. The DTS will be 0 when the MSC starts to send the data frames ongoing. |
| **1**<br>**CTS** | RO | Command Frames Need to Be Sent<br>This bit is set when the command register TCCOR is written, which indicates there are command frames that wait to be sent. The CTS will be 0 when the MSC starts to send the command frames ongoing. |
| **0**<br>**WM** | RW | Work Mode<br>0b - Triggered mode<br>1b - Repetition mode |

### 61.7.5 MSC_TCSTR (Transmitting Channel Status Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | CFB | DFB | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | NP | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_TCSTR field descriptions**

| Field | Type | Description |
|---|---|---|
| **25**<br>**CFB** | RO | Command Frame Busy<br>Indicates if a command frame is currently sent out. |
| **24**<br>**DFB** | RO | Data Frame Busy<br>Indicates if a data frame is currently sent out. |
| **11-8**<br>**NP** | RW | Number of PTFs<br>This bit indicates the number of passive time frames that are inserted in repetition mode between two data frames. |

### 61.7.6 MSC_TCDAR (Transmitting Channel Data Register)

Offset : 1ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TDH | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TDL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_TCDAR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-16**<br>**TDH** | RW | Contains the data bits to be transmitted during the SRH active phase of a data frame. |
| **15-0**<br>**TDL** | RW | Contains the data bits to be transmitted during the SRL active phase of a data frame. |

### 61.7.7 MSC_TCCOR (Transmitting Channel Command Register)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TCH | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TCL | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_TCCOR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-16<br>TCH** | RW | Contains the data bits to be transmitted during the SRH active phase of a command frame. |
| **15-0<br>TCL** | RW | Contains the data bits to be transmitted during the SRL active phase of a command frame. |

### 61.7.8 MSC_TCSLR (Transmitting Channel Source Selection Low Register)

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SSL15 | | | | SSL14 | | | | SSL13 | | | | SSL12 | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SSL11 | | | | SSL10 | | | | SSL9 | | | | SSL8 | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SSL7 | | | | SSL6 | | | | SSL5 | | | | SSL4 | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SSL3 | | | | SSL2 | | | | SSL1 | | | | SSL0 | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_TCSLR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-30**<br>**SSL15** | RW | 00b - SSL[15] is taken from TCDAR.DL[15].<br>10b - SSL[15] is taken from the ALTINL input line 15.<br>11b - SSL[15] is taken from the ALTINL input line 15 in inverted state. |
| **29-28**<br>**SSL14** | RW | 00b - SSL[14] is taken from TCDAR.DL[14].<br>10b - SSL[14] is taken from the ALTINL input line 14.<br>11b - SSL[14] is taken from the ALTINL input line 14 in inverted state. |
| **27-26**<br>**SSL13** | RW | 00b - SSL[13] is taken from TCDAR.DL[13].<br>10b - SSL[13] is taken from the ALTINL input line 13.<br>11b - SSL[13] is taken from the ALTINL input line 13 in inverted state. |
| **25-24**<br>**SSL12** | RW | 00b - SSL[12] is taken from TCDAR.DL[12].<br>10b - SSL[12] is taken from the ALTINL input line 12.<br>11b - SSL[12] is taken from the ALTINL input line 12 in inverted state. |
| **23-22**<br>**SSL11** | RW | 00b - SSL[11] is taken from TCDAR.DL[11].<br>10b - SSL[11] is taken from the ALTINL input line 11.<br>11b - SSL[11] is taken from the ALTINL input line 11 in inverted state. |
| **21-20**<br>**SSL10** | RW | 00b - SSL[10] is taken from TCDAR.DL[10].<br>10b - SSL[10] is taken from the ALTINL input line 10.<br>11b - SSL[10] is taken from the ALTINL input line 10 in inverted state. |
| **19-18**<br>**SSL9** | RW | 00b - SSL[9] is taken from TCDAR.DL[9].<br>10b - SSL[9] is taken from the ALTINL input line 9.<br>11b - SSL[9] is taken from the ALTINL input line 9 in inverted state. |
| **17-16**<br>**SSL8** | RW | 00b - SSL[8] is taken from TCDAR.DL[8].<br>10b - SSL[8] is taken from the ALTINL input line 8.<br>11b - SSL[8] is taken from the ALTINL input line 8 in inverted state. |
| **15-14**<br>**SSL7** | RW | 00b - SSL[7] is taken from TCDAR.DL[7].<br>10b - SSL[7] is taken from the ALTINL input line 7.<br>11b - SSL[7] is taken from the ALTINL input line 7 in inverted state. |
| **13-12**<br>**SSL6** | RW | 00b - SSL[6] is taken from TCDAR.DL[6].<br>10b - SSL[6] is taken from the ALTINL input line 6.<br>11b - SSL[6] is taken from the ALTINL input line 6 in inverted state. |
| **11-10**<br>**SSL5** | RW | 00b - SSL[5] is taken from TCDAR.DL[5].<br>10b - SSL[5] is taken from the ALTINL input line 5.<br>11b - SSL[5] is taken from the ALTINL input line 5 in inverted state. |
| **9-8**<br>**SSL4** | RW | 00b - SSL[4] is taken from TCDAR.DL[4].<br>10b - SSL[4] is taken from the ALTINL input line 4.<br>11b - SSL[4] is taken from the ALTINL input line 4 in inverted state. |

| | | |
|---|---|---|
| **7-6**<br>**SSL3** | RW | 00b - SSL[3] is taken from TCDAR.DL[3].<br>10b - SSL[3] is taken from the ALTINL input line 3.<br>11b - SSL[3] is taken from the ALTINL input line 3 in inverted state. |
| **5-4**<br>**SSL2** | RW | 00b - SSL[2] is taken from TCDAR.DL[2].<br>10b - SSL[2] is taken from the ALTINL input line 2.<br>11b - SSL[2] is taken from the ALTINL input line 2 in inverted state. |
| **3-2**<br>**SSL1** | RW | 00b - SSL[1] is taken from TCDAR.DL[1].<br>10b - SSL[1] is taken from the ALTINL input line 1.<br>11b - SSL[1] is taken from the ALTINL input line 1 in inverted state. |
| **1-0**<br>**SSL0** | RW | 00b - SSL[0] is taken from TCDAR.DL[0].<br>10b - SSL[0] is taken from the ALTINL input line 0.<br>11b - SSL[0] is taken from the ALTINL input line 0 in inverted state. |

### 61.7.9 MSC_TCSHR (Transmitting Channel Source Selection High Register)

Offset : 28h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SSH15 | | SSH14 | | SSH13 | | SSH12 | | SSH11 | | SSH10 | | SSH9 | | SSH8 | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SSH7 | | SSH6 | | SSH5 | | SSH4 | | SSH3 | | SSH2 | | SSH1 | | SSH0 | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_TCSHR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-30**<br>**SSH15** | RW | 00b - SSH[15] is taken from TCDAR.DH[15].<br>10b - SSH[15] is taken from the ALTINH input line 15.<br>11b - SSH[15] is taken from the ALTINH input line 15 in inverted state. |
| **29-28**<br>**SSH14** | RW | 00b - SSH[14] is taken from TCDAR.DH[14].<br>10b - SSH[14] is taken from the ALTINH input line 14.<br>11b - SSH[14] is taken from the ALTINH input line 14 in inverted state. |
| **27-26**<br>**SSH13** | RW | 00b - SSH[13] is taken from TCDAR.DH[13].<br>10b - SSH[13] is taken from the ALTINH input line 13.<br>11b - SSH[13] is taken from the ALTINH input line 13 in inverted state. |

| | | |
|---|---|---|
| **25-24**<br>**SSH12** | RW | 00b - SSH[12] is taken from TCDAR.DH[12].<br>10b - SSH[12] is taken from the ALTINH input line 12.<br>11b - SSH[12] is taken from the ALTINH input line 12 in inverted state. |
| **23-22**<br>**SSH11** | RW | 00b - SSH[11] is taken from TCDAR.DH[11].<br>10b - SSH[11] is taken from the ALTINH input line 11.<br>11b - SSH[11] is taken from the ALTINH input line 11 in inverted state. |
| **21-20**<br>**SSH10** | RW | 00b - SSH[10] is taken from TCDAR.DH[10].<br>10b - SSH[10] is taken from the ALTINH input line 10.<br>11b - SSH[10] is taken from the ALTINH input line 10 in inverted state. |
| **19-18**<br>**SSH9** | RW | 00b - SSH[9] is taken from TCDAR.DH[9].<br>10b - SSH[9] is taken from the ALTINH input line 9.<br>11b - SSH[9] is taken from the ALTINH input line 9 in inverted state. |
| **17-16**<br>**SSH8** | RW | 00b - SSH[8] is taken from TCDAR.DH[8].<br>10b - SSH[8] is taken from the ALTINH input line 8.<br>11b - SSH[8] is taken from the ALTINH input line 8 in inverted state. |
| **15-14**<br>**SSH7** | RW | 00b - SSH[7] is taken from TCDAR.DH[7].<br>10b - SSH[7] is taken from the ALTINH input line 7.<br>11b - SSH[7] is taken from the ALTINH input line 7 in inverted state. |
| **13-12**<br>**SSH6** | RW | 00b - SSH[6] is taken from TCDAR.DH[6].<br>10b - SSH[6] is taken from the ALTINH input line 6.<br>11b - SSH[6] is taken from the ALTINH input line 6 in inverted state. |
| **11-10**<br>**SSH5** | RW | 00b - SSH[5] is taken from TCDAR.DH[5].<br>10b - SSH[5] is taken from the ALTINH input line 5.<br>11b - SSH[5] is taken from the ALTINH input line 5 in inverted state. |
| **9-8**<br>**SSH4** | RW | 00b - SSH[4] is taken from TCDAR.DH[4].<br>10b - SSH[4] is taken from the ALTINH input line 4.<br>11b - SSH[4] is taken from the ALTINH input line 4 in inverted state. |
| **7-6**<br>**SSH3** | RW | 00b - SSH[3] is taken from TCDAR.DH[3].<br>10b - SSH[3] is taken from the ALTINH input line 3.<br>11b - SSH[3] is taken from the ALTINH input line x in inverted state. |
| **5-4**<br>**SSH2** | RW | 00b - SSH[2] is taken from TCDAR.DH[2].<br>10b - SSH[2] is taken from the ALTINH input line 2.<br>11b - SSH[2] is taken from the ALTINH input line 2 in inverted state. |
| **3-2**<br>**SSH1** | RW | 00b - SSH[1] is taken from TCDAR.DH[1].<br>10b - SSH[1] is taken from the ALTINH input line 1.<br>11b - SSH[1] is taken from the ALTINH input line 1 in inverted state. |

| | | |
|---|---|---|
| **1-0**<br>**SSH0** | RW | 00b - SSH[0] is taken from TCDAR.DH[0].<br>10b - SSH[0] is taken from the ALTINH input line 0.<br>11b - SSH[0] is taken from the ALTINH input line 0 in inverted state. |

### 61.7.10 MSC_TCELR (Transmitting Channel Emergency Load Register)

Offset : 2ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EHE15 | EHE14 | EHE13 | EHE12 | EHE11 | EHE10 | EHE9 | EHE8 | EHE7 | EHE6 | EHE5 | EHE4 | EHE3 | EHE2 | EHE1 | EHE0 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ELE15 | ELE14 | ELE13 | ELE12 | ELE11 | ELE10 | ELE9 | ELE8 | ELE7 | ELE6 | ELE5 | ELE4 | ELE3 | ELE2 | ELE1 | ELE0 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_TCELR field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**EHE15** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **30**<br>**EHE14** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **29**<br>**EHE13** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **28**<br>**EHE12** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **27**<br>**EHE11** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **26**<br>**EHE10** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **25**<br>**EHE9** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **24**<br>**EHE8** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |

| | | |
|---|---|---|
| **23**<br>**EHE7** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **22**<br>**EHE6** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **21**<br>**EHE5** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **20**<br>**EHE4** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **19**<br>**EHE3** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **18**<br>**EHE2** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **17**<br>**EHE1** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **16**<br>**EHE0** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDH is used for the shift register load operation. |
| **15**<br>**ELE15** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |
| **14**<br>**ELE14** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |
| **13**<br>**ELE13** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |
| **12**<br>**ELE12** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |
| **11**<br>**ELE11** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |
| **10**<br>**ELE10** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |
| **9**<br>**ELE9** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |
| **8**<br>**ELE8** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |
| **7**<br>**ELE7** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |
| **6**<br>**ELE6** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |

| | | |
|---|---|---|
| **5**<br>**ELE5** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |
| **4**<br>**ELE4** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |
| **3**<br>**ELE3** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |
| **2**<br>**ELE2** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |
| **1**<br>**ELE1** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |
| **0**<br>**ELE0** | RW | If the emergency load feature is enabled for bit x, TCDAR.TDL is used for the shift register load operation. |

### 61.7.11 MSC_RDR0 (Reception Data Register0)

Offset : 30h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RERR | | PF | LAF | | | PB | VLD |
| W | | | | | | | | | | | | | | CLR | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RDATA | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_RDR0 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-22**<br>**RERR** | RO | Reception Error Flag<br>This bit indicates if a parity error, or stop bit error occurred during frame reception.<br>00b - No error<br>01b - Parity error detected<br>10b - Stop bit error detected<br>11b - Both two errors detected |

| | | |
|---|---|---|
| **21**<br>**PF** | RO | Calculated Parity Bit<br>This bit contains the parity bit that has been calculated in the MSC during data frame reception. |
| **20-19**<br>**LAF** | RO | Lower Address Field<br>It contains the two address bits A[1:0] of the 4-bit address field. If 12-bit data frame is selected, LAF is always set to 00. |
| **18**<br>**CLR** | WO | Clear Bit<br>This bit is always read as 0.<br>0b - No operation<br>1b - Bit VLD is cleared. |
| **17**<br>**PB** | RO | Parity Bit<br>This flag contains the parity bit that has been received with the data frame. |
| **16**<br>**VLD** | RO | Valid Bit<br>This bit is set by hardware when the received data is written to RDRx. Writing bit CLR = 1 clears VLD. If hardware setting and software clearing of the valid bit occur simultaneously, the bit VLD will be cleared. |
| **7-0**<br>**RDATA** | RO | Received Data<br>It contains the 8-bit received data. |

### 61.7.12 MSC_RDR1 (Reception Data Register1)

Offset : 34h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RERR | | PF | LAF | | | PB | VLD |
| W | | | | | | | | | | | | | | CLR | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RDATA | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_RDR1 field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **23-22**<br>**RERR** | RO | Reception Error Flag<br>This bit indicates if a parity error, or stop bit error occurred during frame reception.<br>00b - No error<br>01b - Parity error detected<br>10b - Stop bit error detected<br>11b - Both two errors detected |
| **21**<br>**PF** | RO | Calculated Parity Bit<br>This bit contains the parity bit that has been calculated in the MSC during data frame reception. |
| **20-19**<br>**LAF** | RO | Lower Address Field<br>It contains the two address bits A[1:0] of the 4-bit address field. If 12-bit data frame is selected, LAF is always set to 00. |
| **18**<br>**CLR** | WO | Clear Bit<br>This bit is always read as 0.<br>0b - No operation<br>1b - Bit VLD is cleared. |
| **17**<br>**PB** | RO | Parity Bit<br>This flag contains the parity bit that has been received with the data frame. |
| **16**<br>**VLD** | RO | Valid Bit<br>This bit is set by hardware when the received data is written to RDRx. Writing bit CLR = 1 clears VLD. If hardware setting and software clearing of the valid bit occur simultaneously, the bit VLD will be cleared. |
| **7-0**<br>**RDATA** | RO | Received Data<br>It contains the 8-bit received data. |

### 61.7.13 MSC_RDR2 (Reception Data Register2)

Offset : 38h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RERR | | PF | LAF | | | PB | VLD |
| W | | | | | | | | | | | | | | CLR | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RDATA | | | | | | | |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_RDR2 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-22**<br>**RERR** | RO | Reception Error Flag<br>This bit indicates if a parity error, or stop bit error occurred during frame reception.<br>00b - No error<br>01b - Parity error detected<br>10b - Stop bit error detected<br>11b - Both two errors detected |
| **21**<br>**PF** | RO | Calculated Parity Bit<br>This bit contains the parity bit that has been calculated in the MSC during data frame reception. |
| **20-19**<br>**LAF** | RO | Lower Address Field<br>It contains the two address bits A[1:0] of the 4-bit address field. If 12-bit data frame is selected, LAF is always set to 00. |
| **18**<br>**CLR** | WO | Clear Bit<br>This bit is always read as 0.<br>0b - No operation<br>1b - Bit VLD is cleared. |
| **17**<br>**PB** | RO | Parity Bit<br>This flag contains the parity bit that has been received with the data frame. |
| **16**<br>**VLD** | RO | Valid Bit<br>This bit is set by hardware when the received data is written to RDRx. Writing bit CLR = 1 clears VLD. If hardware setting and software clearing of the valid bit occur simultaneously, the bit VLD will be cleared. |
| **7-0**<br>**RDATA** | RO | Received Data<br>It contains the 8-bit received data. |

### 61.7.14 MSC_RDR3 (Reception Data Register3)

Offset : 3ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RERR | | PF | LAF | | | PB | VLD |
| W | | | | | | | | | | | | | | CLR | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RDATA | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_RDR3 field descriptions**

| Field | Type | Description |
|---|---|---|
| **23-22**<br>**RERR** | RO | Reception Error Flag<br>This bit indicates if a parity error, or stop bit error occurred during frame reception.<br>00b - No error<br>01b - Parity error detected<br>10b - Stop bit error detected<br>11b - Both two errors detected |
| **21**<br>**PF** | RO | Calculated Parity Bit<br>This bit contains the parity bit that has been calculated in the MSC during data frame reception. |
| **20-19**<br>**LAF** | RO | Lower Address Field<br>It contains the two address bits A[1:0] of the 4-bit address field. If 12-bit data frame is selected, LAF is always set to 00. |
| **18**<br>**CLR** | WO | Clear Bit<br>This bit is always read as 0.<br>0b - No operation<br>1b - Bit VLD is cleared. |
| **17**<br>**PB** | RO | Parity Bit<br>This flag contains the parity bit that has been received with the data frame. |
| **16**<br>**VLD** | RO | Valid Bit<br>This bit is set by hardware when the received data is written to RDRx. Writing bit CLR = 1 clears VLD. If hardware setting and software clearing of the valid bit occur simultaneously, the bit VLD will be cleared. |
| **7-0**<br>**RDATA** | RO | Received Data<br>It contains the 8-bit received data. |

### 61.7.15 MSC_INCR (Interrupt Control Register)

Offset : 40h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RFIE | | 0 | | TFIE | 0 | | | CFIE | 0 | | | DFIE | | 0 | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_INCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **15-14**<br>**RFIE** | RW | Receive Frame Interrupt Enable<br>00b - Interrupt generation disabled<br>01b - The interrupt is generated when data is received and written into the RDRx<br>10b - The interrupt is generated when received data is not equal to 0<br>11b - The interrupt is generated when data is received and written into register RDR3 |
| **11**<br>**TFIE** | RW | Time Frame Interrupt Enable<br>0b - Disable<br>1b - Enable |
| **7**<br>**CFIE** | RW | Command Frame Interrupt Enable<br>0b - Disable<br>1b - Enable |
| **3-2**<br>**DFIE** | RW | Data Frame Interrupt Enable<br>00b - Disable<br>01b - Interrupt is generated when the last data bit has been shifted<br>10b - Interrupt is generated when the first data bit has been shifted |

### 61.7.16 MSC_INSR (Interrupt Status Register)

Offset : 44h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | RSBFI | RFI | TFI | CFI | DFI |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_INSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **4**<br>**RSBFI** | RO | Receive Data Frame Interrupt Flag when start bit error occurs |
| **3**<br>**RFI** | RO | Receive Data Frame Interrupt Flag |
| **2**<br>**TFI** | RO | Time Frame Interrupt Flag |
| **1**<br>**CFI** | RO | Command Frame Interrupt Flag |
| **0**<br>**DFI** | RO | Data Frame Interrupt Flag |

### 61.7.17 MSC_ISCR (Interrupt Set/Clear Register)

Offset : 48h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | 0 | | | | | | | | | 0 | | | | | |
| W | CRSBFI | | | | | | | | | SDIS | | SDTS | SRFI | STFI | SCFI | SDFI |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | CDIS | CCTS | CDTS | CRFI | CTFI | CCFI | CDFI |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_ISCR field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31**<br>**CRSBFI** | WO | Clear Receive Data Frame Interrupt Flag<br>0b - No operation<br>1b - Clear INSR.RSBFI |
| **22**<br>**SDIS** | WO | Set Disable Transmitting Channel<br>0b - No operation<br>1b - Set TCCTR.TCDIS, which means disable transmitting channel |
| **20**<br>**SDTS** | WO | Set Data Frame Needs to Be Sent<br>0b - No operation<br>1b - Set TCCTR.DTS |
| **19**<br>**SRFI** | WO | Set Receive Data Frame Interrupt Flag<br>0b - No operation<br>1b - Set INSR.RFI |
| **18**<br>**STFI** | WO | Set Time Frame Interrupt Flag<br>0b - No operation<br>1b - Set INSR.TFI |
| **17**<br>**SCFI** | WO | Set Command Frame Interrupt Flag<br>0b - No operation<br>1b - Set INSR.CFI |
| **16**<br>**SDFI** | WO | Set Data Frame Interrupt Flag<br>0b - No operation<br>1b - Set INSR.DFI |
| **6**<br>**CDIS** | WO | Clear Disable Transmitting Channel<br>0b - No operation<br>1b - Clear TCCTR.TCDIS, which means enable transmitting channel |
| **5**<br>**CCTS** | WO | Clear Command Frame Needs to Be Sent<br>0b - No operation<br>1b - Clear TCCTR.CTS |
| **4**<br>**CDTS** | WO | Clear Data Frame Needs to Be Sent<br>0b - No operation<br>1b - Clear TCCTR.DTS |
| **3**<br>**CRFI** | WO | Clear Receive Data Frame Interrupt Flag<br>0b - No operation<br>1b - Clear INSR.RFI |
| **2**<br>**CTFI** | WO | Clear Time Frame Interrupt Flag<br>0b - No operation<br>1b - Clear INSR.TFI |

| | | |
|---|---|---|
| **1**<br>**CCFI** | WO | Clear Command Frame Interrupt Flag<br>0b - No operation<br>1b - Clear INSR.CFI |
| **0**<br>**CDFI** | WO | Clear Data Frame Interrupt Flag<br>0b - No operation<br>1b - Clear INSR.DFI |

### 61.7.18 MSC_IOCR (Input/Output Control Register)

Offset : 4ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | IDS | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | ESC | | ESH | | ESL | | FCLC TRL | 0 | | | | IPS | ENP | SOP | FCLP |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_IOCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **18-16**<br>**IDS** | RW | Input Data Selection<br>000 - SDI0 input is selected for SDI.<br>001 - SDI1 input is selected for SDI.<br>010 - SDI2 input is selected for SDI.<br>011 - SDI3 input is selected for SDI.<br>100 - SDI4 input is selected for SDI.<br>101 - SDI5 input is selected for SDI.<br>110 - SDI6 input is selected for SDI.<br>111 - SDI7 input is selected for SDI. |
| **14-13**<br>**ESC** | RW | Enable Selection for ENC<br>Selects the enable Enx that becomes active during ENC = 1 of a command frame.<br>00b - EN0 is selected for ENC.<br>01b - EN1 is selected for ENC.<br>10b - EN2 is selected for ENC.<br>11b - EN3 is selected for ENC. |

| | | |
|---|---|---|
| **12-11**<br>**ESH** | RW | Enable Selection for ENH<br>Selects the enable Enx that becomes active during SRH active phase of a data frame.<br>00b - EN0 is selected for ENH.<br>01b - EN1 is selected for ENH.<br>10b - EN2 is selected for ENH.<br>11b - EN3 is selected for ENH. |
| **10-9**<br>**ESL** | RW | Enable Selection for ENL<br>Select the chip enable output Enx that becomes active during the SRL active phase of a data frame.<br>00b - EN0 is selected for ENL.<br>01b - EN1 is selected for ENL.<br>10b - EN2 is selected for ENL.<br>11b - EN3 is selected for ENL. |
| **8**<br>**FCLCTRL** | RW | FCL Control<br>0b - FCL is active only during the active phases of frames.<br>1b - FCL is always active whether or not a downstream frame is transmitted currently. |
| **3**<br>**IPS** | RW | Input Polarity Selection<br>0b - SDI and SI signal polarities are identical.<br>1b - SDI and SI signal polarities are inverted. |
| **2**<br>**ENP** | RW | ENx Polarity<br>0b - Enx and ENL, ENH, ENC signal polarities are identical (high active).<br>1b - ENx and ENL, ENH, ENC signal polarities are inverted (low active). |
| **1**<br>**SOP** | RW | SO Polarity<br>0b - SOP and SO signal polarity are identical.<br>1b - SOP and SO signal polarity are inverted. |
| **0**<br>**FCLP** | RW | FCL Polarity<br>0b - FCLP and FCL polarity are identical.<br>1b - FCLP and FCL polarity are inverted. |

### 61.7.19 MSC_TCCTR1 (Transmitting Channel Control Register1)

Offset : 58h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | 0 | INJP1 | | | | | | INJE1 | 0 | INJP0 | | | | | | INJE0 |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

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

**MSC_TCCTR1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **30-25**<br>**INJP1** | RW | Injection Position 1<br>This bit indicates the position of the injected signal bit1. |
| **24**<br>**INJE1** | RW | Injection Enable 1<br>This bit is injection enable of INJ[1]. |
| **22-17**<br>**INJP0** | RW | Injection Position 0<br>This bit indicates the position of the injected signal bit0. |
| **16**<br>**INJE0** | RW | Injection Enable 0<br>This bit is injection enable of INJ[0]. |

### 61.7.20 MSC_RTOR (Reception Timeout Register)

Offset : 5ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | TOF | TOIE |
| W | | | | | | | | | | | | | TOFC | TOFS | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TOV | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_RTOR field descriptions**

| Field | Type | Description |
|---|---|---|
| **19**<br>**TOFC** | WO | Receiving Timeout Flag Clear<br>0b - No action<br>1b - Clear timeout flag |

| | | |
|---|---|---|
| **18**<br>**TOFS** | WO | Receiving Timeout Flag Set<br>0b - No action<br>1b - Set timeout flag |
| **17**<br>**TOF** | RO | Timeout Flag<br>Signal an upstream timer event. |
| **16**<br>**TOIE** | RW | Receiving Timeout Interrupt Enable<br>0b - Disable timeout interrupt<br>1b - Enable timeout interrupt |
| **15-0**<br>**TOV** | RW | Timeout Value<br>Receiving timeout value (internal timeout value = TOV*bit time) |

### 61.7.21 MSC_SRCR (Software Reset Clear Register)

Offset : ech

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | 0 |
| W | | | | | | | | | | | | | | | | RCLR |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_SRCR field descriptions**

| Field | Type | Description |
|---|---|---|
| **0**<br>**RCLR** | WO | Clear Software Reset Done Flag<br>0b - No action<br>1b - Clear MSR.RDONE flag |

### 61.7.22 MSC_MSR (Software Reset Register)

Offset : f4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | RDONE | RST |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSC_MSR field descriptions**

| Field | Type | Description |
|---|---|---|
| **1**<br>**RDONE** | RO | Software Reset Done<br>This bit can be cleared by writing 1 to the RCLR bit in SRCR register.<br>The reset done can only be asserted after the current reception and transmission are done.<br>1b - Software reset has been finished<br>0b - No software reset was executed |
| **0**<br>**RST** | RW | Software Reset<br>All registers except MSR and SRCR, and functions in transmitting and reception channels will be reset.<br>0b - No operation<br>1b - Reset is request |

Flagchip

# Chapter 62 Single Edge Nibble Transmission (SENT)

## 62.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| 0.2 | 2024-01-15 | • Added a new subsection *SPC Overrun* to *Section SPC Mode;*<br>• Added a new bit field *SPC_OVERRUN* to the register section *SENT_CHN_STATn;*<br>• Added a new bit field *SPC_OVERRUN_EN* to the register section *SENT_CHN_CTLn* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

This chapter describes the Single Edge Nibble Transmission (SENT) Module. It contains the following sections:

- Chip-specific Information
- Overview
- Features
- Functional Description
- Register Definition

## 62.2 Chip-specific Information

N/A

## 62.3 Overview

The SENT is a Single Edge Nibble Transmission encoding scheme which conforms to SENT SAE J2716 2010 standard. It is intended for use in applications where high-resolution sensor data needs to be communicated from a sensor to an Electronic Control Unit (ECU). Also, it is used as a replacement for the lower resolution methods of 10-bit A/D's and PWM and as a simpler low-cost alternative to CAN or LIN. The implementation assumes that the sensor is a smart sensor containing a microprocessor or dedicated logic device (ASIC) to create the signal.

In addition to standard protocols, the SENT also supports Short PWM Code (SPC) mode. In this mode, the sensor needs to receive a coordination signal from the controller and then start transmitting the messages. Apart from this mode, the SENT is a unidirectional communications scheme from the sensor to the controller which does not include a coordination signal from the controller. The sensor signal is transmitted as a series of pulses with data encoded as falling to falling edge periods.

### 62.3.1 Modes of Operation

The SENT supports operation in Run Mode, Debug Mode, and Wait mode of the chip:

**Table 116. SENT modes of operation**

| Mode | Description |
|---|---|
| Run | Normal operation |
| Wait | Can continue operating in Wait mode and generate an interrupt to wake up the chip |
| Stop/Standby | Stop working under these modes |
| Debug | Can continue operating in Debug mode when the Debug Enable bit (SENT_CLBL_CTL [dbg_en]) is set. |

## 62.4 Features

The SENT supports the following features:

- Support independent receiving of 4-channels data
- Input signal digital filter
- Configurable receive data nibble number
- Configurable receive nibble data mode from H1 to H7
- Calibration pulse compensation function to support clock tick time with a variation of 20%
- Detection of nibble value errors
- Detection of falling edge number errors
- Detection of sync/calibration pulse difference errors
- Detection of calibration diagnosis errors
- Detection of CRC errors
- Two options available for successive calibration pulses diagnosis
- Optional pause pulse in one frame

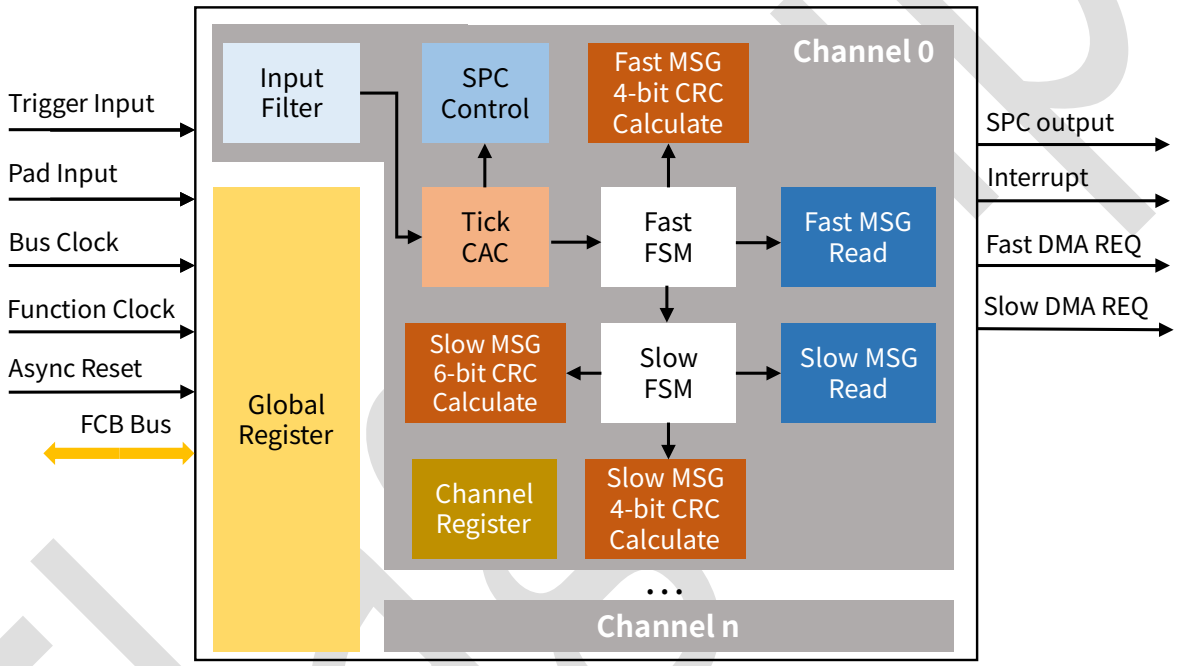
- Support 76-bit*16 FIFO for each channel
- Support Fast Message, Short Serial Message, and Enhanced Serial Message
- Support SPC mode which supports software triggers or triggers from SOC

## 62.5 Block diagram

The Single Edge Nibble Transmission module provides a serial communication link that is targeted for high-resolution data transmitted from a sensor to the ECU. The SENT has four independent channels; each channel has a dedicated fast logic, slow logic, digital filter, and SPC Control logic. The SENT can perform as a unidirectional communication that only receives the data from the sensor. It can also perform as a bidirectional communication that needs the SENT to output an SPC Pulse (a synchronization signal) first and then start the communication.

The block diagram and interface are shown in the below figure.

**Figure 142. SENT block diagram**




## 62.6 Functional Description

The SENT supports four transfer types: Fast Message, Short Serial Message, Enhanced Serial Message, and SPC Mode. Since the Short Serial Message and Enhanced Serial Message use the same register to store received data, the data will be referred to as "slow message" uniformly.

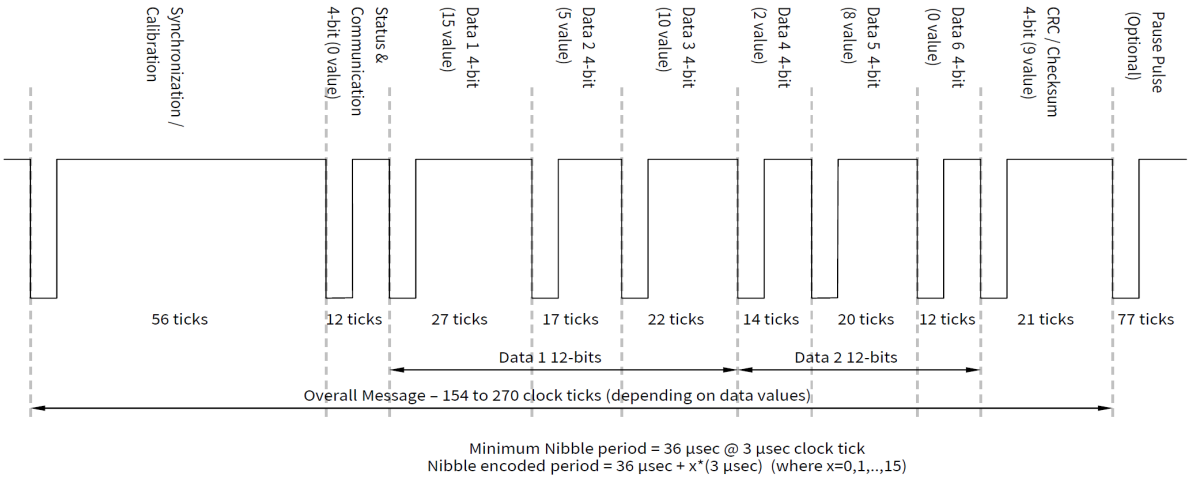
### 62.6.1 Message Definition

The SENT encoding scheme consists of a sequence of pulses that is repeatedly sent by the transmitting module. One basic frame consists of the following sequence:

- Calibration/Synchronization pulse period of 56 clock ticks.
- One 4-bit Status and Serial Communication nibble pulse of 12 to 27 clock ticks.
- A sequence of one up to six 4-bit Data nibble pulses (12 to 27 clock ticks each) representing the values of the signal(s) to be communicated. The number of nibbles will be fixed for each application of the encoding scheme (i.e., throttle position sensors, mass air flow, etc.) but can vary between applications. For example, if two 12-bit values are transmitted, 6 nibbles will be communicated.
- One 4-bit Checksum nibble pulse of 12 to 27 clock ticks.
- One optional pause pulse.

The figure below shows the encoding scheme of the fast message with 6 data nibbles. The clock tick is 3 microseconds. Note that the shortest length message, based on a valid checksum consists of 154 clock ticks (Data nibbles [0 0 0 1 1 0] and Checksum 0, Data nibbles [0 0 1 0 0 0] and Checksum 1, and Data nibbles [1 0 0 0 0 1] and Checksum 0). Similarly, the longest message is 270 clock ticks (Data nibbles [$E $F $F $F $F $E] and Checksum $F, Data nibbles [$F $F $E $F $F $F] and Checksum $E, and Data nibbles [$F $F $F $E $E $F] and Checksum $F).

**Figure 143. Encoding scheme for 6 data fast message**




Signals conveyed using the Data nibbles are also referred to as "Fast Channel" signals with SENT specification.

#### *62.6.1.1 Calibration/Synchronization Pulse*

The Calibration/Synchronization Pulse has the following features:

- Nominal pulse period is 56 clock ticks.
- Duration of being driven low is of more than 4 clock ticks (all remaining clock ticks – driven high).
- Actual period measured by receiving module to correct for a transmitter clock variation.

#### *62.6.1.2 CRC/Checksum nibble*

The Checksum nibble is a 4-bit CRC of the Data nibbles only. The Status and Communication Nibble is not included in the CRC calculation by default. Configuring CHN_CTL[FCRC_SC_EN] to 1 can include the Status and Communication Nibble into CRC calculation.

The CRC is calculated using polynomial x^4 + x^3 + x^2 + 1 with a seed value of 0101. For CRC generation, the data nibbles will be augmented by a nibble with data = 0.

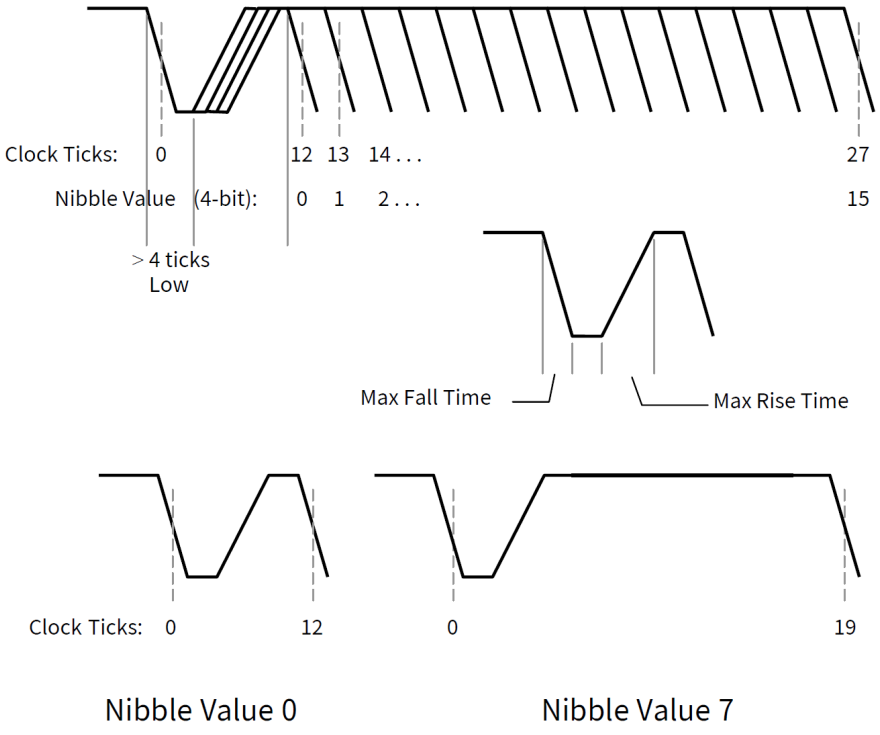
#### *62.6.1.3 Nibble Pulse*

The Nibble Pulse has the following features:

- The minimum pulse period is 12 clock ticks.
- Duration of being driven low is of more than 4 clock ticks (all remaining clock ticks – driven high).
- Each nibble count is 1 clock tick (0 - 15 counts 0 - 45 µs at a 3 µs clock tick).
- Minimum nibble pulse period (transmission value of 0) = 12 clock ticks (36 µs at a 3 µs clock tick).
- Maximum nibble pulse period (transmission value of 15) = 12 + 15 = 27 clock ticks (36 + 45 = 81 µs at a 3 µs clock tick).

The figure below shows the pictorial representation of some nibble properties.

**Figure 144. Example nominal nibble times**




#### *62.6.1.4 Status and Communication Nibble*

This nibble is reserved to enable the sensor to transmit miscellaneous information such as part numbers or error code information. The nibble is defined in the following table:

**Table 117. Status and communication nibble description**

| Bit Number | Function |
|---|---|
| 0 (least significant) | Reserved for a specific application |
| 1 | Reserved for a specific application |
| 2 | Used in the short serial message and enhanced serial message |
| 3 | |

Bits 2 and 3 define a transmitter optional serial message channel which can be implemented either as a short serial message format or an enhanced serial message format as specified below. SENT receivers shall support both the short serial message format as well as the enhanced serial message format. The Enhanced Serial Message Format is recommended rather than the Short Serial Message Format.

### 62.6.2 Standard Mode

In SENT SAE J2716 2010 standard, only Fast Message Format, Short Serial Message Format, and Enhanced Serial Message Format are supported. In these standard modes, transmission occurs independently of any action of the receiver module. The SENT acts as a unidirectional communications scheme from the sensor to the controller (SOC) which does not include a coordination signal from the controller.

#### *62.6.2.1 Fast Message Format*

The scheme of Fast Message Format is shown in *Figure 143. Encoding scheme for 6 data fast message.* It includes a synchronization pulse, up to 6 nibbles and an optional pause pulse. Each Fast Message contains independent information from the sensor.

Usually, the Fast Message is transmitted from the sensor continuously, and the Synchronization Pulse of the messages will be used to diagnose the correctness of the received message.

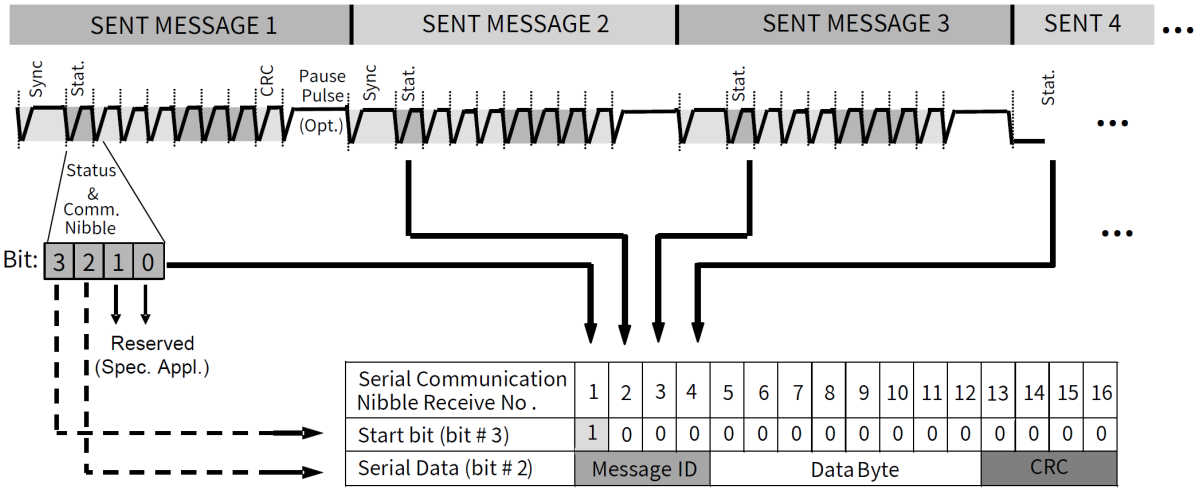
#### 62.6.2.2 Short Serial Message Format

Serial data is transmitted (bit by bit) in bit 2 (of the Status and Communication nibble) of consecutive messages from the transmitter. Serial data will be communicated in a 16-bit sequence as shown in the figure below. The starting bit of a serial message is indicated by a "1" in bit 3 of the Status and Communication nibble. The next 15 received frames must contain a value of "0" in this same bit 3 position, as shown in the figure below. All 16 frames must be successfully received (no errors, calibration pulse variation, data nibble CRC error, etc.) for the serial value to be received (the figure below).

The 16-bit message consists of a 4-bit Message ID, a Data Byte, and a 4-bit CRC checksum. The CRC checksum is derived for the Message ID and Data Byte and is the same checksum algorithm as used to calculate the SENT CRC nibble. The Message ID is used to identify the type of data being communicated in the Data Byte. Actual Serial Data message IDs and data values are application specific.

All data transmitted in the Serial Data Bit (bit #2) is sent in the order Most Significant Bit to Least Significant Bit.

**Figure 145. Construction of short serial data message from 16 SENT messages**



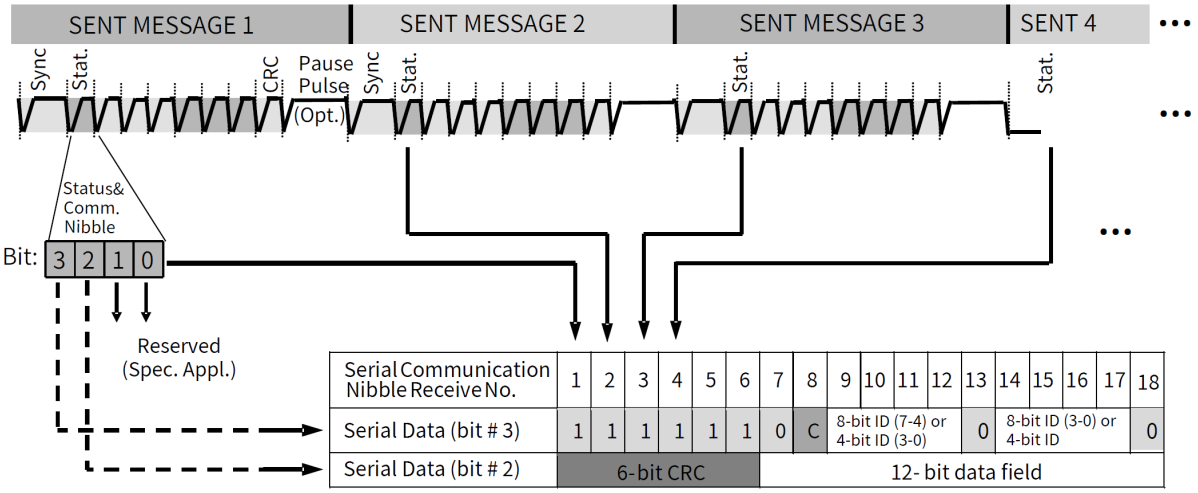

One serial message is composed of 16 SENT consecutive error-fee messages.

#### 62.6.2.3 Enhanced Serial Message Format

An optional enhanced serial message format can be used by sensors, which require a serial communication channel with a larger data field and a larger set of message IDs. If the enhanced serial message format is used, serial data is transmitted in bit #2 and bit #3 of the status and communication nibble. A serial message frame stretches over 18 consecutive SENT data messages from the transmitter as shown in the figure below. All 18 frames must be successfully received (no errors, calibration pulse variation, data nibble CRC error, etc.) for the serial value to be received.

The table in the figure below defines the frame format. The frame start of a serial message is indicated by the unique pattern "01111110" in bit 3 of the status and communication nibble (SENT message #18, #1 to #7). The first "1" in a series of six ones (after a "0") indicates the start of a serial message frame. The "0" in SENT message #13 is defined to ensure the uniqueness of the start pattern. At initialization, it is recommended that the transmitter send the Serial data bit #3 sequence "0···01111110" (start with one or more SENT messages equivalent to #18). One starting "0" will guarantee the minimum latency start sequence and no interrupted message (due to sensor reset) will appear to be completed and incorrectly pass the CRC.

**Figure 146. Construction of enhanced serial data message from 18 SENT messages**




One serial message is composed of 18 SENT consecutive error-fee messages.

The serial message frame contains 20 bits of payload data. Two different configurations can be chosen determined by the configuration bit (serial data bit #3, serial communication nibble No. 8):

- 12-bit data and 8-bit message ID (configuration bit = 0)
- 16-bit data and 4-bit message ID (configuration bit = 1)

All data (data field, message ID and CRC) that is transmitted in the serial message channel is sent in the order MSB (most significant bit) to LSB (least significant bit). The mapping and the order of the data bits, the message ID, the configuration bit, and the CRC is detailed in *Figure 147. Enhanced serial message format with 12-bit data field and 8-bit message ID* or the configuration 12-bit data + 8-bit ID and in *Figure 148. Enhanced serial message format with 16-bit data field and 4-bit message ID* for the configuration 16-bit data and 4-bit ID.

**Figure 147. Enhanced serial message format with 12-bit data field and 8-bit message ID**

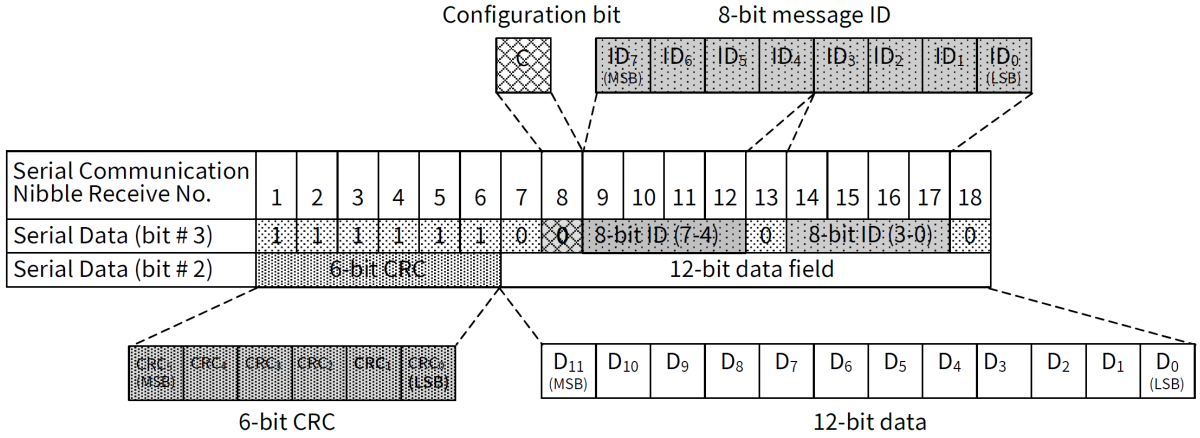
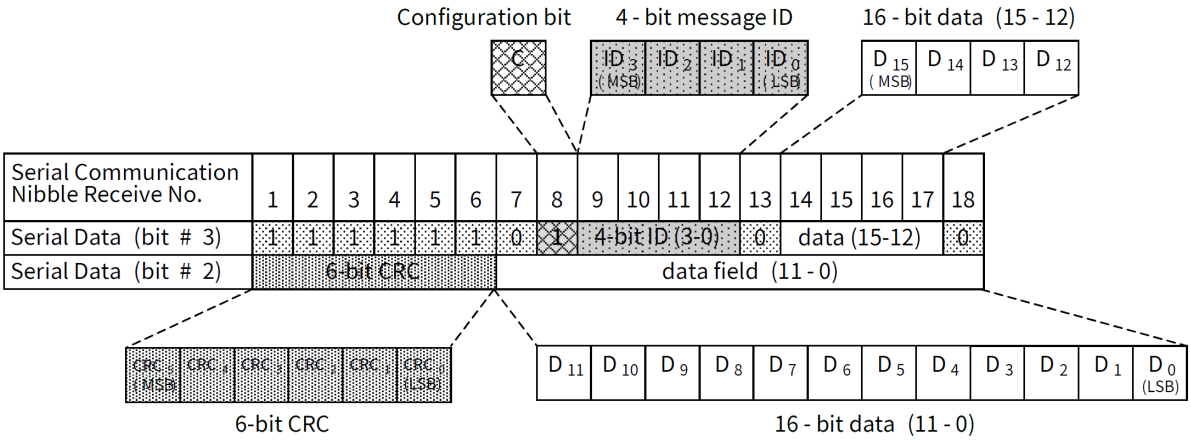


**Figure 148. Enhanced serial message format with 16-bit data field and 4-bit message ID**



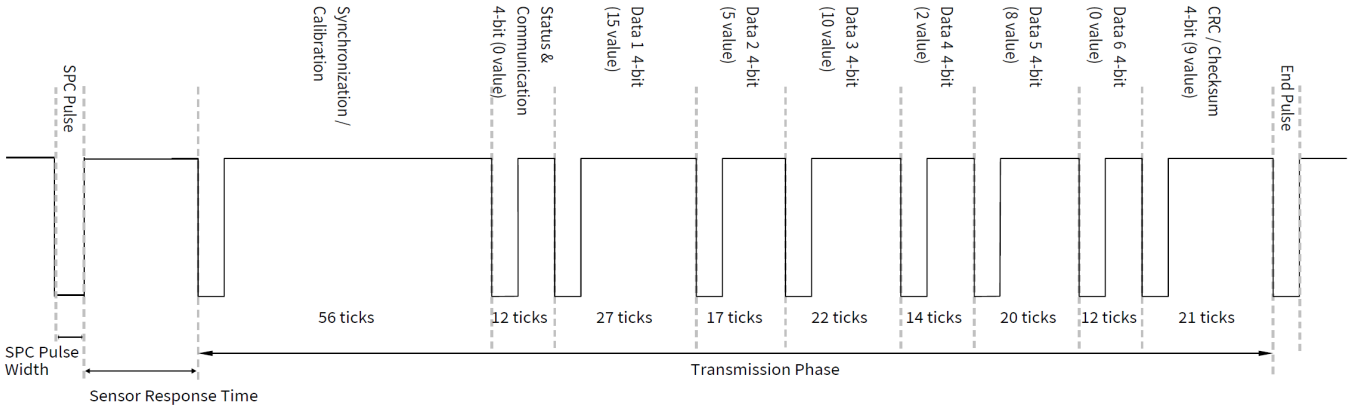

### 62.6.3 SPC Mode

The SENT also supports SPC mode. This mode enhances the SENT protocol defined by the SAE 2716 specification. In this mode, the transmission occurs depending on an SPC Pulse (a synchronization signal) from the receiver module, and the SPC Pulse is implemented as a low pulse. When the sensor receives the coordination signal from SOC and then it starts transmitting the messages.

#### *62.6.3.1 SPC Message Format*

Different from the fast message format, the first pulse in SPC Message format is the SPC Pulse and then followed by the Sync Pulse. And there is no Pause Pulse in SPC mode. Instead, an End Pulse is appended at the end of the message to indicate that the message is finished. The figure below shows the message format of SPC mode.

**Figure 149. SPC message format**




The width of an SPC Pulse can be configured by writing the register CHN_SPC_CTL[SPC_PULSE_WIDTH] and the width range is from 1 tick to 64 ticks. After the sensor receives the SPC Pulse, it needs a response time to recognize the synchronization signal and prepare the transmission message; the time value depends on the sensor. The sensor starts transmitting the message after it is ready, the transmission phase sequence is the same as the fast message format. After transmitting the CRC nibble, the sensor should give an extra low pulse to indicate the end of this message and then pull up the signal to wait for the next SPC Pulse trigger.

After the SENT outputs one SPC Pulse, it cannot output another SPC Pulse until it receives one correct message from the sensor. If the sensor cannot feedback the message for an unexpectedly long time, the SPC Pulse output function can be reset by writing the register CHN_SPC_CTL[SPC_ENABLE] to 0, then the SENT can be configured to output the SPC Pulse again.

#### *62.6.3.2 SPC Pulse Generation Methods*

To generate an SPC Pulse, two methods are implemented: external trigger mode and software mode.

- For external trigger mode, the SENT has an input to receive trigger sources from TRGSEL in SOC.
  a) Configure the SENT SPC function to external trigger mode by writing the register CHN_SPC_CTL[SPC_MODE] to 1.
  b) Set CHN_SPC_CTL[SPC_PULSE_WIDTH] and CHN_SPC_CTL[SPC_PULSE_DELAY].
  c) Enable CHN_SPC_CTL[SPC_ENABLE], then the SENT will output one SPC Pulse when receiving one trigger from TRGSEL.
- For software mode:
  a) Configure the register CHN_SPC_CTL[SPC_MODE] to 0.
  b) Set CHN_SPC_CTL[SPC_PULSE_WIDTH] and CHN_SPC_CTL[SPC_PULSE_DELAY].
  c) Enable CHN_SPC_CTL[SPC_ENABLE], then the SENT will output one SPC Pulse when setting CHN_SPC_CTL[SPC_START].

#### *62.6.3.3 SPC Overrun*

If a new SPC request asserts while SENT is handling a SPC request, a SPC Overrun Flag will assert to indicate that the new SPC request will not be executed. Read register CHN_STAT[SPC_OVERRUN] for the SPC overrun flag. If CHN_CTL [SPC_OVERRUN_IE] is enable, an interrupt will assert when the SPC overrun flag is set.

### 62.6.4 CRC Calculation

Two CRC calculation algorithms are supported for Fast Message Format, SPC Message Format and Short Serial Message Format: SAE J2716 2016 CRC checksum algorithm (SAE CRC) and alternative CRC checksum algorithm (Alt. CRC).

In alternative CRC checksum algorithm, the CRC will be calculated using a polynomial x^4 + x^3 + x^2 + 1 with a seed value of 0101. The checksum nibble will be a 4-bit data as well. To select Alt. CRC, enable CHN_CTL[ALT_4BIT_CRC]. This alternative CRC checksum algorithm is applicable for some mainstream sensors, for example, Infineon TLE4998.

For both Alt. CRC and SAE CRC, the S&C nibble can be selected into CRC calculation by configuring the register CHN_CTL[FCRC_SC_EN]. This option is only for Fast Message Format and SPC Message Format.

For CRC generation, the data nibbles will be augmented by a nibble with data = 0 by default. The augmentation can be excluded from CRC generation by enabling the register CHN_CTL[FCRC_SEL] for Fast Message/SPC Message and CHN_CTL[SCRC_SEL] for Short Serial Message.

Below example is an implementation of the alternative CRC checksum algorithm:

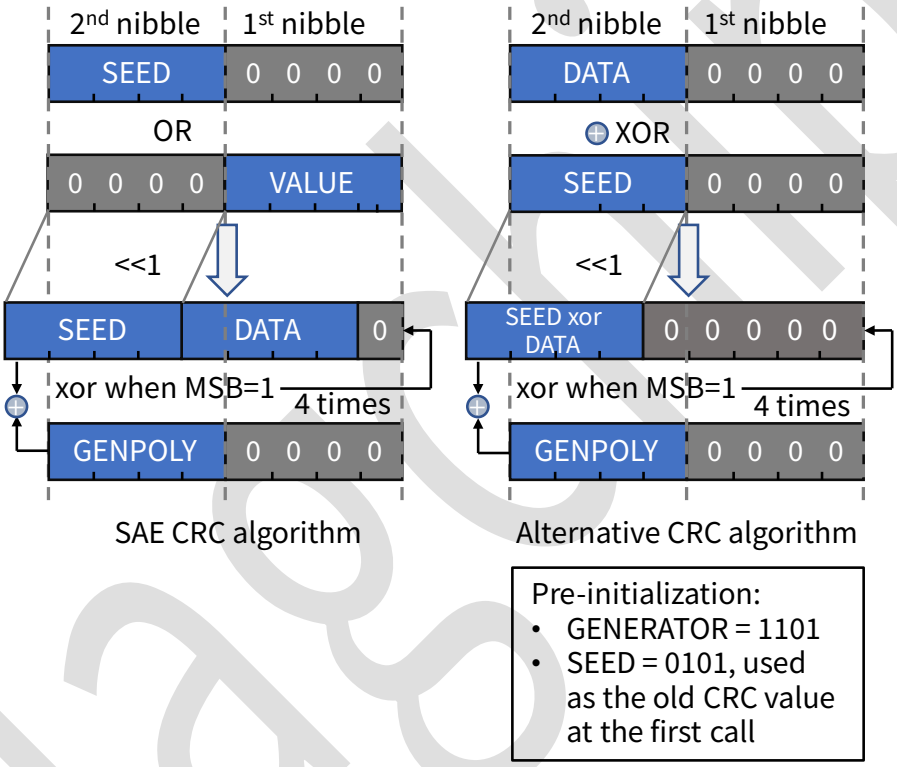
```
lookup_table = [0,13,7,10,14,3,9,4,1,12,6,11,15,2,8,5];
checksum0 =5;
// if enable CHN_CTL [FCRC_SC_EN]
data = [S&C nibble, d1, d2, d3, d4, d5, d6];
for (i=0; i<7; i++) {
   checksum0 = checksum0 ^ data[i];
    checksum0 = lookup_table[checksum0];
}
// if disable CHN_CTL [FCRC_SC_EN]
```

```
data = [d1, d2, d3, d4, d5, d6];
for (i=0; i<6; i++) {
  checksum0 = checksum0 ^ data[i];
   checksum0 = lookup_table[checksum0];
}
```

Check checksum0 with CRC nibble after calculation finished.

The figure below shows the comparison of SAE CRC algorithm with Alt CRC algorithm:

**Figure 150. Comparison of SAE CRC algorithm and Alt CRC algorithm**




## 62.7 Initialization

This section describes setup steps of different messages of the SENT.

### 62.7.1 Fast Message

The Fast Message can be set up by the below steps:

1) Configure CHN_CLK_CTL register to set the PRE_SCALER value. This value is used to generate a tick clock.
2) Configure CHN_CTL register to enable a specific interrupt source and set other options (i.e., PAUSE_EN).
3) Configure DATA_NUM_CTL1 register to set the channel data nibble number.
4) Configure CHN_EN register to enable the selected channel.
5) Configure GLBL_CTL register to enable SENT_EN and other fields.
6) Configure FINT_CTL register to enable a fast message ready interrupt.
7) Start receiving a Fast Message and waiting for a SENT fast message ready flag, then read the fast message from the registers CHN_FDATA, and CHN_FCRC.

### 62.7.2 Short Serial Message

The Short Serial Message can be set up by the below steps:

1) Configure CHN_CLK_CTL register to set the PRE_SCALER value. This value is used to generate a tick clock.
2) Configure CHN_CTL register to enable a specific interrupt source and set other options (i.e., PAUSE_EN).
3) Configure DATA_NUM_CTL1 register to set the channel data nibble number.
4) Configure CHN_EN register to enable the selected channel.
5) Configure GLBL_CTL register to enable SENT_EN and other fields.
6) Configure SINT_CTL register to enable a slow message ready interrupt.
7) Start receiving a Serial Message and waiting for a SENT serial message ready flag, then read the serial message from the registers CHN_SBIT3, CHN_SBIT2, and CHN_STS.

### 62.7.3 Enhanced Serial Message

The Enhanced Serial Message configuration is the same as Short Serial Message.

### 62.7.4 SPC MODE

The SPC Pulse can be generated by an external trigger or software.

#### *62.7.4.1 SPC External Trigger Mode*

The SPC External Trigger Mode can be set up by the steps below:

1) Configure CHN_CLK_CTL register to set the PRE_SCALER value. This value is used to generate a tick clock.
2) Configure CHN_CTL register to enable a specific interrupt source, disable PAUSE_EN, set SCP_CHK to 0 and set other options.
3) Configure DATA_NUM_CTL1 register to set the channel data nibble number.
4) Configure CHN_EN register to enable the selected channel.
5) Configure GLBL_CTL register to enable SENT_EN and other fields.
6) Configure CHN_SPC_CTL to set SPC_PULSE_WIDTH value, SPC_PULSE_DELAY value, and select SPC_TICK_BASE.
7) Configure CHN_SPC_CTL to select SPC_MODE to external trigger mode (SPC_MODE = 1), set CAL_DIAG_DIS optionally.
8) Configure CHN_SPC_CTL to enable SPC_ENABLE bit.
9) External trigger input will trigger the SENT to output an SPC Pulse.
10) After receiving a correct message. The fast message ready flag will assert and get the received data in the registers CHN_FDATA, and CHN_FCRC.

#### *62.7.4.2 SPC Software Mode*

The SPC Software Trigger Mode can be set up by the steps below:

1) Configure CHN_CLK_CTL register to set the PRE_SCALER value. This value is used to generate A tick clock.
2) Configure CHN_CTL register to enable specific interrupt source, disable PAUSE_EN, set SCP_CHK to 0 and set other options.
3) Configure DATA_NUM_CTL1 register to set the channel data nibble number.
4) Configure CHN_EN register to enable the selected channel.
5) Configure GLBL_CTL register to enable SENT_EN and other fields.
6) Configure CHN_SPC_CTL to set SPC_PULSE_WIDTH value, SPC_PULSE_DELAY value, and select SPC_TICK_BASE.
7) Configure CHN_SPC_CTL to select SPC_MODE to software trigger mode (SPC_MODE = 0), and set CAL_DIAG_DIS optionally.
8) Configure CHN_SPC_CTL to enable SPC_ENABLE bit.
9) Enable CHN_SPC_CTL[SPC_START] to trigger the SENT to output an SPC Pulse.
10) After receiving a correct message. The fast message ready flag will assert, and get the received data in the registers CHN_FDATA, and CHN_FCRC.

### 62.7.5 Fast Message DMA Transfer

The Fast Message DMA Transfer can be set up by the below steps:

1) Configure CHN_CLK_CTL register to set the PRE_SCALER value. This value is used to generate a tick clock.
2) Configure CHN_CTL register to enable a specific interrupt source and set other options (i.e., PAUSE_EN).
3) Configure DATA_NUM_CTL1 register to set the channel data nibble number.
4) Configure CHN_EN register to enable the selected channel.
5) Configure GLBL_CTL register to enable SENT_EN and other fields.
6) Configure FDMA_CTL register to enable a fast message DMA request.
7) Configure DMA to read a fast message from register CHN_DFD, and CHN_DFC.
8) Start receiving the Fast Message. The Fast DMA request will assert when one correct message is received.

### 62.7.6 Fast Message DMA Transfer with FIFO Enable

The Fast Message DMA Transfer with FIFO Enable can be set up by the below steps:

1) Configure CHN_CLK_CTL register to set the PRE_SCALER value. This value is used to generate a tick clock.
2) Configure CHN_CTL register to enable a specific interrupt source and set other options (i.e., PAUSE_EN).
3) Configure DATA_NUM_CTL1 register to set the channel data nibble number.
4) Configure CHN_EN register to enable the selected channel.
5) Configure GLBL_CTL register to enable SENT_EN, set DMA_WATERMARK and other fields.
6) Configure FIFO_EN register to enable FIFO.
7) Configure FDMA_CTL register to enable a fast message DMA request.
8) Configure DMA to read a fast message from the registers CHN_DFD, and CHN_DFC.
9) Start receiving the Fast Message. The Fast DMA request will assert if the received message number is greater than the watermark.

### 62.7.7 Short/Enhanced Serial Message DMA Transfer

The Short/Enhanced Serial message DMA Transfer can be set up by the below steps:

1) Configure CHN_CLK_CTL register to set the PRE_SCALER value. This value is used to generate a tick clock.
2) Configure CHN_CTL register to enable a specific interrupt source and set other options (i.e., PAUSE_EN).
3) Configure DATA_NUM_CTL1 register to set the channel data nibble number.
4) Configure CHN_EN register to enable the selected channel.
5) Configure GLBL_CTL register to enable SENT_EN and other fields.
6) Configure SDMA_CTL register to enable a slow message DMA request.
7) Configure DMA to read a serial message from the registers CHN_SBIT3, CHN_SBIT2, and CHN_STS.
8) Start receiving the Serial Message. The Slow (serial) DMA request will assert when one correct serial message is received.

Flagchip

## 62.8 Register Definition

SENT base address : 4005_4000h

**Register Overview - SENT**

| Address (hex) | Register Name | Width (bits) | Access | Reset Value | Page |
|---|---|---|---|---|---|
| **0h** | SENT_GLBL_CTL (Global Control Register) | 32 | RW | 0000_0010h | page |
| **4h** | SENT_CHN_EN (Channel Enable Register) | 32 | RW | 0000_0000h | page |
| **8h** | SENT_GLBL_STAT (Global Status Register) | 32 | W1C | 0000_0000h | page |
| **Ch** | SENT_FMSG_STAT (Fast Message Status Register) | 32 | RW | 0000_0000h | page |
| **10h** | SENT_SMSG_STAT (Slow Message Status Register) | 32 | RW | 0000_0000h | page |
| **14h** | SENT_FIFO_EN (FIFO Enable Register) | 32 | RW | 0000_0000h | page |
| **18h** | SENT_DATA_NUM_CTL1 (Each Channel Data Number Control Register) | 32 | RW | 1111_0000h | page |
| **20h** | SENT_OVFL_UDFL_STAT (Fast Message FIFO Overflow DMA Underflow State Register) | 32 | RW | 0000_0000h | page |
| **24h** | SENT_NB_MODE1 (Channel Nibble Data Mode Control 1 Register) | 32 | RW | 0000_0000h | page |
| **2Ch** | SENT_FDMA_CTL (Channel Fast Message DMA Control Register) | 32 | RW | 0000_0000h | page |
| **30h** | SENT_SDMA_CTL (Channel Slow Message DMA Control Register) | 32 | RW | 0000_0000h | page |
| **34h** | SENT_FINT_CTL (Channel Fast Message Interrupt Control Register) | 32 | RW | 0000_0000h | page |
| **38h** | SENT_SINT_CTL (Channel Slow Message Interrupt Control Register) | 32 | RW | 0000_0000h | page |
| **3Ch+n*50h** | SENT_CHN_CLK_CTLn (Channel Clock Control Register) | 32 | RW | 0000_8000h | page |
| **40h+n*50h** | SENT_CHN_STATn (Channel Status Register) | 32 | W1C | 0000_0000h | page |
| **44h+n*50h** | SENT_CHN_CTLn (Channel Control Register) | 32 | RW | 0000_0104h | page |
| **48h+n*50h** | SENT_CHN_FDATAn (Channel Received Fast Data Nibble Value Register) | 32 | RO | 0000_0000h | page |
| **4Ch+n*50h** | SENT_CHN_FCRCn (Channel Fast Message CRC and Status & Communication Value Register) | 32 | RO | 0000_0000h | page |
| **50h+n*50h** | SENT_CHN_FTSn (Channel Fast Timestamp Value Register) | 32 | RO | 0000_0000h | page |
| **54h+n*50h** | SENT_CHN_SBIT3_n (Channel Slow Message Status & Communication Bit 3 Received Value Register) | 32 | RO | 0000_0000h | page |
| **58h+n*50h** | SENT_CHN_SBIT2_n (Channel Slow Message Status & Communication Bit 2 Received Value Register) | 32 | RO | 0000_0000h | page |

| | | | | | |
|---|---|---|---|---|---|
| **5Ch+n*50h** | SENT_CHN_STSn (Channel Slow Message Timestamp Value Register) | 32 | RW | 0000_0000h | page |
| **60h+n*50h** | SENT_CHN_DFDn (Channel n DMA Fast Data Register) | 32 | RW | 0000_0000h | page |
| **64h+n*50h** | SENT_CHN_DFCn (Channel n DMA Fast CRC Data Register) | 32 | RW | 0000_0000h | page |
| **68h+n*50h** | SENT_CHN_DFTSn (Channel n DMA Fast Timestamp Register) | 32 | RW | 0000_0000h | page |
| **70h+n*50h** | SENT_CHN_DSB3_n (Channel n DMA Slow Message Bit3 Register) | 32 | RW | 0000_0000h | page |
| **74h+n*50h** | SENT_CHN_DSB2_n (Channel n DMA Slow Message Bit2 Register) | 32 | RW | 0000_0000h | page |
| **78h+n*50h** | SENT_CHN_DSTSn (Channel n DMA Slow Message Timestamp Register) | 32 | RW | 0000_0000h | page |
| **7Ch+n*50h** | SENT_CHN_TSTMPn (Channel n Timestamp Register) | 32 | RW | 0000_0000h | page |
| **80h+n*50h** | SENT_CHN_MCNTn (Channel n Message Counter Register) | 32 | RW | 0000_0000h | page |
| **84h+n*50h** | SENT_CHN_SPC_CTLn (Channel n SPC Mode Control Register) | 32 | RW | 0000_0000h | page |
| **88h+n*50h** | SENT_CHN_IDLE_CTLn (Channel n Idle Control Register) | 32 | RW | 0000_0000h | page |

### 62.8.1 SENT_GLBL_CTL (Global Control Register)

Offset : 0h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PRE_SCALER | | | | | | | | 0 | | | DMA_WATERMARK | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | F_FIFO_OVFL_IE | F_DMA_UDFL_IE | S_DMA_UDFL_IE | 0 | | | FAST_CLR | 0 | DBG_EN | 0 | SENT_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**SENT_GLBL_CTL field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-24**<br>**PRE_SCALER** | RW | Time Stamp Prescaler Counter<br>This value is used to divide the function clock to count the timestamp. |
| **20-16**<br>**DMA_WATERMARK** | RW | DMA Watermark |

| | | |
|---|---|---|
| **10**<br>**F_FIFO_OVFL_IE** | RW | Fast Message FIFO Overflow Interrupt Enable<br>Global control register of fast message FIFO overflow interrupt generate.<br>0b - Fast message FIFO overflow interrupt disabled.<br>1b - Fast message FIFO overflow interrupt enabled. |
| **9**<br>**F_DMA_UDFL_IE** | RW | Fast Message DMA Read Underflow Interrupt Enable<br>0b - Fast message DMA underflow interrupt disabled.<br>1b - Fast message DMA underflow interrupt enabled. |
| **8**<br>**S_DMA_UDFL_IE** | RW | Slow Message DMA Read Underflow Interrupt Enable<br>0b - Slow message DMA underflow interrupt disabled.<br>1b - Slow message DMA underflow interrupt enabled. |
| **4**<br>**FAST_CLR** | RW | Data Overflow Flag Fast Clear<br>When enabled after data ready read of received data, CRC and timestamp register will prevent the assert of data overflow flag.<br>0b - Fast clear disabled.<br>1b - Fast clear enabled. |
| **2**<br>**DBG_EN** | RW | Debug Mode Enable<br>0b - SENT will stop running under debug mode.<br>1b - SENT will keep running under debug mode. |
| **0**<br>**SENT_EN** | RW | Global SENT Enabled Control<br>When enable SENT receive channel, this bit must be enabled.<br>0b - SENT receive disabled for all channels.<br>1b - SENT receive enabled. |

### 62.8.2 SENT_CHN_EN (Channel Enable Register)

Offset : 4h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | CHN_EN | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_EN field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **3-0**<br>**CHN_EN** | RW | Channel Receive Enable<br>Each channel enables control register.<br>0001b - Channel 0 enabled.<br>0010b - Channel 1 enabled.<br>0100b - Channel 2 enabled.<br>1000b - Channel 3 enabled. |

### 62.8.3 SENT_GLBL_STAT (Global Status Register)

Offset : 8h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | S_DMA_UDFL | | | |
| W | | | | | | | | | | | | | w1c | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_GLBL_STAT field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**S_DMA_UDFL** | W1C | Channel Slow Message DMA Read Underflow Flag<br>It means related channel slow message DMA read underflow happens when it asserts.<br>0001b - Channel 0 slow message DMA underflow.<br>0010b - Channel 1 slow message DMA underflow.<br>0100b - Channel 2 slow message DMA underflow.<br>1000b - Channel 3 slow message DMA underflow. |

### 62.8.4 SENT_FMSG_STAT (Fast Message Status Register)

Offset : Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | F_MSG_RDY | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_FMSG_STAT field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**F_MSG_RDY** | RW | Channel Fast Message Ready Flag<br>When assert, it means related channel fast message is ready for read out.<br>0001b - Channel 0 fast message ready.<br>0010b - Channel 1 fast message ready.<br>0100b - Channel 2 fast message ready.<br>1000b - Channel 3 fast message ready. |

### 62.8.5 SENT_SMSG_STAT (Slow Message Status Register)

Offset : 10h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | S_MSG_RDY | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_SMSG_STAT field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**S_MSG_RDY** | RW | Channel Slow Message Ready Flag<br>When assert, it means related channel slow message is ready for read out.<br>0001b - Channel 0 slow message ready.<br>0010b - Channel 1 slow message ready.<br>0100b - Channel 2 slow message ready.<br>1000b - Channel 3 slow message ready. |

### 62.8.6 SENT_FIFO_EN (FIFO Enable Register)

Offset : 14h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | FIFO_EN | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_FIFO_EN field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**FIFO_EN** | RW | Channel FIFO Control<br>0001b - Channel 0 FIFO enable.<br>0010b - Channel 1 FIFO enable.<br>0100b - Channel 2 FIFO enable.<br>1000b - Channel 3 FIFO enable. |

### 62.8.7 SENT_DATA_NUM_CTL1 (Each Channel Data Number Control Register)

Offset : 18h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | CH0_DATA_CTL | | | 0 | CH1_DATA_CTL | | | 0 | CH2_DATA_CTL | | | 0 | CH3_DATA_CTL | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_DATA_NUM_CTL1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **30-28**<br>**CH0_DATA_CTL** | RW | Channel 0 Data Number Control<br>The received data number will be configured value plus 1, 0 and value greater than 5 is forbidden. |
| **26-24**<br>**CH1_DATA_CTL** | RW | Channel 1 Data Number Control<br>The received data number will be configured value plus 1, 0 and value greater than 5 is forbidden. |
| **22-20**<br>**CH2_DATA_CTL** | RW | Channel 2 Data Number Control<br>The received data number will be configured value plus 1, 0 and value greater than 5 is forbidden. |
| **18-16**<br>**CH3_DATA_CTL** | RW | Channel 3 Data Number Control<br>The received data number will be configured value plus 1, 0 and value greater than 5 is forbidden. |

### 62.8.8 SENT_OVFL_UDFL_STAT (Fast Message FIFO Overflow DMA Underflow State Register)

Offset : 20h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | FIFO_F_OVFL_FLAG | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | DMA_F_UDFL_FLAG | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_OVFL_UDFL_STAT field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16**<br>**FIFO_F_OVFL_FLAG** | RW | Channel Fast Message FIFO Overflow Flag<br>Bit n of this register field represents the state of channel n.<br>0b - Fast message FIFO not overflow.<br>1b - Fast message FIFO overflow happen. |
| **3-0**<br>**DMA_F_UDFL_FLAG** | RW | Channel Fast Message DMA Underflow Flag<br>Bit n of this register field represents the state of channel n<br>0b - Fast message channel DMA read not underflow.<br>1b - Fast message channel DMA read underflow happens. |

### 62.8.9 SENT_NB_MODE1 (Channel Nibble Data Mode Control 1 Register)

Each field in this register controls different channel and has the same function. Configure this field will change the receive data order arrangement in register. Value 0 will not reorder the received data, value 1 to 7 will reorder the fast channel1 and fast channel 2 data according to data formats in SENT SPEC SAE J2716 201604 appendix H.1 to H.7

Offset : 24h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CH0_NB_MODE | | | | CH1_NB_MODE | | | | CH2_NB_MODE | | | | CH3_NB_MODE | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_NB_MODE1 field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-28**<br>**CH0_NB_MODE** | RW | Channel0 Nibble Data Mode Control<br>(Refer to SENT SAE J2716 2010 for detail protocol definition)<br>0001b - Frame format H1<br>0010b - Frame format H2<br>0011b - Frame format H3<br>0100b - Frame format H4<br>0101b - Frame format H5<br>0110b - Frame format H6<br>0111b - Frame format H7 |
| **27-24**<br>**CH1_NB_MODE** | RW | Channel1 Nibble Data Mode Control<br>(Refer to SENT SAE J2716 2010 for detail protocol definition)<br>0001b - Frame format H1<br>0010b - Frame format H2<br>0011b - Frame format H3<br>0100b - Frame format H4<br>0101b - Frame format H5<br>0110b - Frame format H6<br>0111b - Frame format H7 |

| | | |
|---|---|---|
| **23-20**<br>**CH2_NB_MODE** | RW | Channel2 Nibble Data Mode Control<br>(Refer to SENT SAE J2716 2010 for detail protocol definition)<br>0001b - Frame format H1<br>0010b - Frame format H2<br>0011b - Frame format H3<br>0100b - Frame format H4<br>0101b - Frame format H5<br>0110b - Frame format H6<br>0111b - Frame format H7 |
| **19-16**<br>**CH3_NB_MODE** | RW | Channel3 Nibble Data Mode Control<br>(Refer to SENT SAE J2716 2010 for detail protocol definition)<br>0001b - Frame format H1<br>0010b - Frame format H2<br>0011b - Frame format H3<br>0100b - Frame format H4<br>0101b - Frame format H5<br>0110b - Frame format H6<br>0111b - Frame format H7 |

### 62.8.10 SENT_FDMA_CTL (Channel Fast Message DMA Control Register)

Offset : 2Ch

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | FDMA_EN |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_FDMA_CTL field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **3-0**<br>**FDMA_EN** | RW | Fast Message DMA Request Enable<br>Bit n in this field controls channel n's fast message DMA request. When FIFO enabled, the request will asserts if data number in FIFO is more than watermark configured in GLBL_CTL[water_mark].<br>0b - Fast message DMA request disabled.<br>1b - Fast message DMA request enabled. |

### 62.8.11 SENT_SDMA_CTL (Channel Slow Message DMA Control Register)

Offset : 30h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | SDMA_EN | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_SDMA_CTL field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**SDMA_EN** | RW | Slow Message DMA Request Enable<br>Bit n in this field controls channel n's slow message DMA request.<br>0b - Slow message DMA request disabled.<br>1b - Slow message DMA request enabled. |

### 62.8.12 SENT_FINT_CTL (Channel Fast Message Interrupt Control Register)

Offset : 34h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | FINT_CTL | | | |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_FINT_CTL field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**FINT_CTL** | RW | Fast Message Interrupt Control<br>Bit n in this register controls the channel n's interrupt generation.<br>0b - Fast message interrupt disabled.<br>1b - Fast message interrupt enabled. |

### 62.8.13 SENT_SINT_CTL (Channel Slow Message Interrupt Control Register)

Offset : 38h

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | SINT_CTL | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_SINT_CTL field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**SINT_CTL** | RW | Slow message interrupt control<br>Bit n in this register controls the channel n's interrupt generation.<br>0b - Slow message interrupt disabled.<br>1b - Slow message interrupt enabled. |

### 62.8.14 SENT_CHN_CLK_CTLn (Channel Clock Control Register)

Offset : 3Ch+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | COMPED_PRE_SCALER | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COMP_EN | 0 | PRE_SCALER | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_CLK_CTLn field descriptions**

| Field | Type | Description |
|---|---|---|
| **30-16**<br>**COMPED_PRE_SCALER** | RO | Prescaler value after compensate |
| **15**<br>**COMP_EN** | RW | Compensate Enable<br>This register field controls the tick period compensate function. If enabled the the receiver will compensate the tick period difference between the sensor and receiver according to the difference of sync&calibration pulse.<br>0b - Compensate disabled.<br>1b - Compensate enabled. |
| **13-0**<br>**PRE_SCALER** | RW | Channel Tick Time Scaler<br>This register field is used to divide the function clock to tick clock. |

### 62.8.15 SENT_CHN_STATn (Channel Status Register)

Offset : 40h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BUS_IDLE | 0 | SPC_OVERRUN | 0 | CAL_RESYNC | CAL_20_25 | S_OVFL | F_OVFL | NVAL_LESS_ERR | PP_DIAG_ERR | CAL_DIAG_ERR | CAL_ERR | NVAL_EXCEED_ERR | S_CRC_ERR | F_CRC_ERR | EDGE_ERR |
| W | w1c | | w1c | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_STATn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**BUS_IDLE** | W1C | Receiver Bus Idle Flag<br>0b - Bus is not idle.<br>1b - Bus is idle. |
| **29**<br>**SPC_OVERRUN** | W1C | SPC Mode Overrun Flag<br>0b - SPC mode is not overrun.<br>1b - SPC mode is overrun. |
| **27**<br>**CAL_RESYNC** | W1C | Continuous 2 Sync/Calibration Pulse Difference Exceed 1.5625% Error Flag<br>0b - cal_resync pass.<br>1b - cal_resync error flag set. |
| **26**<br>**CAL_20_25** | W1C | Calibration 20% Pass 25% Fail Flag<br>0b - cal_20_25 flag not set.<br>1b - cal_20_25 flag set. |
| **25**<br>**S_OVFL** | W1C | Slow Message Overflow Flag<br>0b - Slow message not overflow.<br>1b - Slow message overflow. |
| **24**<br>**F_OVFL** | W1C | Fast Message Overflow Flag<br>0b - Fast message not overflow.<br>1b - Fast message overflow. |
| **23**<br>**NVAL_LESS_ERR** | W1C | Nibble Value Less than Zero Error Flag<br>0b - Nibble value not less than 0.<br>1b - Nibble value is less than 0. |
| **22**<br>**PP_DIAG_ERR** | W1C | Previous Pulse Diagnosis Error Flag<br>0b - pp_diag_err flag cleared<br>1b - pp_diag_err flag set |
| **21**<br>**CAL_DIAG_ERR** | W1C | Calibration Diagnosis over 25% Error Flag<br>0b - cal_diag_err flag cleared.<br>1b - cal_diag_err flag set. |
| **20**<br>**CAL_ERR** | W1C | Sync/calibration Pulse Difference Exceed 1.5625% Error Flag<br>0b - cal_err flag cleared.<br>1b - cal_err flag set. |
| **19**<br>**NVAL_EXCEED_ERR** | W1C | Nibble Value More than 15 Error Flag<br>0b - nval_exceed_err flag cleared.<br>1b - nval_exceed_err flag set. |

| | | |
|---|---|---|
| **18**<br>**S_CRC_ERR** | W1C | Slow Message CRC Check Error Flag<br>0b - s_crc_err flag cleared.<br>1b - s_crc_err flag set. |
| **17**<br>**F_CRC_ERR** | W1C | Fast Message CRC Check Error Flag<br>0b - f_crc_err flag cleared<br>1b - f_crc_err flag set |
| **16**<br>**EDGE_ERR** | W1C | Falling Edge Number Error Flag<br>0b - edge_err flag cleared.<br>1b - edge_err flag set. |

### 62.8.16 SENT_CHN_CTLn (Channel Control Register)

Offset : 44h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BUS_IDLE_IE | 0 | SPC_OVERRUN_IE | ALT_4BIT_CRC | CAL_RESYNC_IE | CAL_20_25_IE | SMSG_OVFL_IE | FMSG_OVFL_IE | FMSG_CRC_CHECK_OFF | PP_DIAG_ERR_IE | CAL_DIAG_ERR_IE | CAL_ERR_IE | NIB_ERR_IE | S_CRC_ERR_IE | F_CRC_ERR_IE | EDGE_ERR_IE |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA_CHANGE_EN | CAL_SEL | PP_CHK_DIS | FCRC_SEL | FCRC_SC_EN | SCRC_SEL | PAUSE_EN | SCP_CHK | FILT_CNT | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**SENT_CHN_CTLn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**BUS_IDLE_IE** | RW | Bus Idle Interrupt Enable |
| **29**<br>**SPC_OVERRUN_IE** | RW | SPC Mode Overrun Interrupt Enable<br>0b - Disable SPC mode overrun interrupt<br>1b - Enable SPC mode overrun interrupt |

| | | |
|---|---|---|
| **28**<br>**ALT_4BIT_CRC** | RW | Alternative 4-bit CRC Algorithm<br>If enabled, the CRC calculation will use alternative 4-bit CRC algorithm instead of SAE CRC algorithm. |
| **27**<br>**CAL_RESYNC_IE** | RW | Continuous 2 Sync/Calibration Pulse Difference Exceed 1.5625% Error Interrupt Enable |
| **26**<br>**CAL_20_25_IE** | RW | Calibration 20% Pass 25% Fail Interrupt Enable |
| **25**<br>**SMSG_OVFL_IE** | RW | Slow Message Overflow Interrupt Enable |
| **24**<br>**FMSG_OVFL_IE** | RW | Fast Message Overflow Interrupt Enable |
| **23**<br>**FMSG_CRC_CHECK_OFF** | RW | Fast Message CRC Check Control<br>0b - Fast message CRC check enable.<br>1b - Fast message CRC check disable. |
| **22**<br>**PP_DIAG_ERR_IE** | RW | Previous Pulse Diagnosis Error Interrupt Enable |
| **21**<br>**CAL_DIAG_ERR_IE** | RW | Calibration diagnosis over 25% Error Interrupt Enable |
| **20**<br>**CAL_ERR_IE** | RW | Sync/Calibration Pulse Difference Exceed 1.5625% Error Interrupt Enable |
| **19**<br>**NIB_ERR_IE** | RW | Nibble Value More than 15 or Less than 0 Error Interrupt Enable |
| **18**<br>**S_CRC_ERR_IE** | RW | Slow Message CRC Check Error Interrupt Enable |
| **17**<br>**F_CRC_ERR_IE** | RW | Fast Message CRC Check Error Interrupt Enable |
| **16**<br>**EDGE_ERR_IE** | RW | Falling Edge Number Error Interrupt Enable |
| **15**<br>**DATA_CHANGE_EN** | RW | Data Change Enable<br>If enabled, the SENT will only assert the fast message ready flag when received data has changed compared with previous data. |
| **14**<br>**CAL_SEL** | RW | Calibration Valid Selection<br>0b - The difference of received calibration pulse and receiver configuration is within 20%, the message is deemed to be valid.<br>1b - The difference of received calibration pulse and receiver configuration is more than 20% but less than 25%, the message is deemed to be valid. |

| | | |
|---|---|---|
| **13**<br>**PP_CHK_DIS** | RW | Successive Calibration Pulses Diagnostic Disable<br>If set, the successive calibration pulses diagnostic will be disabled. |
| **12**<br>**FCRC_SEL** | RW | Fast Message CRC Calculate Method Selection<br>This bit is to select whether the augmentation is added into the CRC calculation. If selected, a 4-bit zero will be added at the end of the CRC calculation.<br>0b - Augmentation is selected for Fast Message CRC calculation.<br>1b - Augmentation is not selected for Fast Message CRC calculation. |
| **11**<br>**FCRC_SC_EN** | RW | Fast Message CRC Calculate with S&C Nibble Enable<br>If enabled, the CRC calculation will include S&C nibble. |
| **10**<br>**SCRC_SEL** | RW | Short Serial Message CRC Calculate Method Selection<br>This bit is to select whether the augmentation is added into the CRC calculation. If selected, a 4-bit zero will be added at the end of the CRC calculation.<br>Enhanced Serial Message CRC calculation is not influenced by this bit and augmentation is selected in default.<br>0b - Augmentation is selected for Short Serial Message CRC calculation.<br>1b - Augmentation is not selected for Short Serial Message CRC calculation. |
| **9**<br>**PAUSE_EN** | RW | Pause Pulse Enable<br>If enabled, a pause pulse will be appended at the end of a frame. |
| **8**<br>**SCP_CHK** | RW | Successive Calibration Pulses Diagnostic Option<br>0b - Diagnostic option2<br>1b - Preferred option |
| **7-0**<br>**FILT_CNT** | RW | Digital Filter Count Value<br>The input signal needs keep stable for (filt_cnt<<1) * function_clk cycles to pass through the digital filter. |

### 62.8.17 SENT_CHN_FDATAn (Channel Received Fast Data Nibble Value Register)

Offset : 48h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA1 | | | | DATA2 | | | | DATA3 | | | | DATA4 | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA5 | | | | DATA6 | | | | RESERVED0 | | | | RESERVED1 | | | |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_FDATAn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-28**<br>**DATA1** | RO | Received Fast Message Data Nibble 1 |
| **27-24**<br>**DATA2** | RO | Received Fast Message Data Nibble 2 |
| **23-20**<br>**DATA3** | RO | Received Fast Message Data Nibble 3 |
| **19-16**<br>**DATA4** | RO | Received Fast Message Data Nibble 4 |
| **15-12**<br>**DATA5** | RO | Received Fast Message Data Nibble 5 |
| **11-8**<br>**DATA6** | RO | Received Fast Message Data Nibble 6 |
| **7-4**<br>**RESERVED0** | RO | |
| **3-0**<br>**RESERVED1** | RO | |

### 62.8.18 SENT_CHN_FCRCn (Channel Fast Message CRC and Status & Communication Value Register)

Offset : 4Ch+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | CRC_DATA | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | SC_NB | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_FCRCn field descriptions**

| Field | Type | Description |
|---|---|---|
| **19-16<br>CRC_DATA** | RO | Received Fast Message CRC Data |
| **3-0<br>SC_NB** | RO | Fast Message Status Communication Nibble Value |

### 62.8.19 SENT_CHN_FTSn (Channel Fast Timestamp Value Register)

Offset : 50h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIMESTAMP_VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIMESTAMP_VAL | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_FTSn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0<br>TIMESTAMP_VAL** | RO | Fast Message Timestamp Value |

### 62.8.20 SENT_CHN_SBIT3_n (Channel Slow Message Status & Communication Bit 3 Received Value Register)

Offset : 54h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CHN_NUM | | | | MSG_TYPE | 0 | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | CFG | ID7_4_OR_ID3_0 | | | | 0 | ID3_0_OR_DATA15_12 | | | | 0 |
| W | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_SBIT3_n field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-28**<br>**CHN_NUM** | RO | Channel Number |
| **27**<br>**MSG_TYPE** | RO | Indicates whether this is a short serial message or an enhanced serial message.<br>0b - Received message is a short serial message.<br>1b - Received message is an enhanced serial message. |
| **10**<br>**CFG** | RO | Configuration Bit in Enhanced Serial Message<br>0b - Enhanced serial message format is 8-bit ID + 12-bit data.<br>1b - Enhanced serial message format is 4-bit ID + 16-bit data. |
| **9-6**<br>**ID7_4_OR_ID3_0** | RO | This field is read as enhanced serial message ID bit 7 to bit 4 when CFG bit is 0; and when CFG bit is 1, this read as ID bit 3 to ID bit 0. |
| **4-1**<br>**ID3_0_OR_DATA15_12** | RO | This field is read as enhanced serial message ID bit 3 to bit 0 when CFG bit is 0; and when CFG bit is 1, this read as data bit 15 to bit 12. |

### 62.8.21 SENT_CHN_SBIT2_n (Channel Slow Message Status & Communication Bit 2 Received Value Register)

Offset : 58h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | SMSG_CRC | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | SMSG_DATA | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_SBIT2_n field descriptions**

| Field | Type | Description |
|---|---|---|
| **21-16**<br>**SMSG_CRC** | RO | Slow Message CRC Data<br>When short serial message received, the 4-bit CRC data is from bit 19 to bit 16. |

| | | |
|---|---|---|
| **11-0**<br>**SMSG_DATA** | RO | Slow Message Data<br>When short serial message received, the data is bit 7 to bit 0. |

### 62.8.22 SENT_CHN_STSn (Channel Slow Message Timestamp Value Register)

Offset : 5Ch+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SMSG_TIMESTAMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SMSG_TIMESTAMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_STSn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**SMSG_TIMESTAMP** | RO | Slow Message Timestamp Value |

### 62.8.23 SENT_CHN_DFDn (Channel n DMA Fast Data Register)

Offset : 60h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FMSG_DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FMSG_DATA | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_DFDn field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-0**<br>**FMSG_DATA** | RO | Fast Message Data Read by DMA |

### 62.8.24 SENT_CHN_DFCn (Channel n DMA Fast CRC Data Register)

Offset : 64h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FMSG_CRC | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FMSG_CRC | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_DFCn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0**<br>**FMSG_CRC** | RO | Fast Message CRC Value Read by DMA |

### 62.8.25 SENT_CHN_DFTSn (Channel n DMA Fast Timestamp Register)

Offset : 68h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FMSG_TSTMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FMSG_TSTMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_DFTSn field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-0 FMSG_TSTMP** | RO | Fast Message Timestamp Value Read by DMA |

### 62.8.26 SENT_CHN_DSB3_n (Channel n DMA Slow Message Bit3 Register)

Offset : 70h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SMSG_BIT3 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SMSG_BIT3 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_DSB3_n field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 SMSG_BIT3** | RO | Slow Message Bit3 Data Read by DMA |

### 62.8.27 SENT_CHN_DSB2_n (Channel n DMA Slow Message Bit2 Register)

Offset : 74h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SMSG_BIT2 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SMSG_BIT2 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_DSB2_n field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-0 SMSG_BIT2** | RO | Slow Message Bit2 Data Read by DMA |

### 62.8.28 SENT_CHN_DSTSn (Channel n DMA Slow Message Timestamp Register)

Offset : 78h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SMSG_TSTMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SMSG_TSTMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_DSTSn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31-0 SMSG_TSTMP** | RO | Slow Message Timestamp Read by DMA |

### 62.8.29 SENT_CHN_TSTMPn (Channel n Timestamp Register)

Offset : 7Ch+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CHN_TSTMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CHN_TSTMP | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_TSTMPn field descriptions**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **31-0 CHN_TSTMP** | RO | Channel n Timestamp |

### 62.8.30 SENT_CHN_MCNTn (Channel n Message Counter Register)

Offset : 80h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | 0 | | | | | | | FRAME_CNT | | | | | | | |
| W | FRAME_CNT_CLR | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | EDGE_CNT | | | | | | | |
| W | EDGE_CNT_CLR | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_MCNTn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31 FRAME_CNT_CLR** | WO | Clear the number of received frames. |
| **23-16 FRAME_CNT** | RO | Indicates the number of received frames. |
| **15 EDGE_CNT_CLR** | WO | Clear the number of received edges. |
| **7-0 EDGE_CNT** | RO | Indicates the number of received edges. |

### 62.8.31 SENT_CHN_SPC_CTLn (Channel n SPC Mode Control Register)

Offset : 84h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SPC_ENAB | 0 | | | | | | | | | | | | | CAL_DIAG | SPC_BUSY |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | LE | | | | | | | | | | | | | | _DIS | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | SPC_MODE | | SPC_PULSE_DELAY | | | | | | SPC_TICK_BASE | 0<br>SPC_START | SPC_PULSE_WIDTH | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_SPC_CTLn field descriptions**

| Field | Type | Description |
|---|---|---|
| **31**<br>**SPC_ENABLE** | RW | This bit is used to enable SPC mode.<br>0b - Clear SPC busy state.<br>1b - Enable SPC mode. |
| **17**<br>**CAL_DIAG_DIS** | RW | Calibration Diagnosis Disable<br>This field is used to disable the calibration diagnosis in SPC mode. |
| **16**<br>**SPC_BUSY** | RO | Indicates the SPC is busy<br>When spc_busy is set, the SENT can not output another SPC pulse until received a correct message. |
| **15-14**<br>**SPC_MODE** | RW | This field is used to select the SPC pulse is triggered by external trigger or by software.<br>00b - SPC pulse triggered by software method.<br>01b - SPC pulse triggered by external trigger. |
| **13-8**<br>**SPC_PULSE_DELAY** | RW | This field is used to configure the delay between SPC-trigger assert and finally generate a SPC pulse.<br>(spc_pulse_delay + 1) * tick as the final SPC pulse delay. |
| **7**<br>**SPC_TICK_BASE** | RW | This field is used to select the tick base of SPC.<br>0b - Previous received message tick base.<br>1b - SENT configured tick base. |
| **6**<br>**SPC_START** | WO | This field is used to trigger the SENT to generate a SPC pulse by software. |
| **5-0**<br>**SPC_PULSE_WIDTH** | RW | This field is used to configure the width of the SPC pulse.<br>(spc_pulse_width + 1) * tick as the final SPC pulse width. |

### 62.8.32 SENT_CHN_IDLE_CTLn (Channel n Idle Control Register)

Offset : 88h+N*50h (N : 0 ~ 3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | IDLE_CNT | | | |
| W | | | | | | | | | | | | | | | | |
| RST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**SENT_CHN_IDLE_CTLn field descriptions**

| Field | Type | Description |
|---|---|---|
| **3-0**<br>**IDLE_CNT** | RW | This field determines how long the bus idle flag will assert when bus is idle.<br>0000b - The bus idle flag will never assert.<br>0001b - The bus is idle for more than 127*2 ticks, then the bus idle flag will assert.<br>0010b - The bus is idle for more than 127*4 ticks, then the bus idle flag will assert.<br>0100b - The bus is idle for more than 127*8 ticks, then the bus idle flag will assert.<br>1000b - The bus is idle for more than 127*16 ticks, then the bus idle flag will assert. |

# Chapter 63 FLASH API

## 63.1 Revision History

| Revision | Date | Changes |
|---|---|---|
| 0.1 | 2023-11-20 | Preliminary version |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Flagchip

## 63.2 Overview

The purpose of this chapter is to explain the architecture and implementation of FLASH API. It provides users with interface functions for operating FLASH, which is convenient for users to apply.

## 63.3 Functional Description

FLASH API provides FLASH erasing (BLOCK/SECTOR/NVR), programming, array integrity check, margin read check, ECC/EDC self-check, watchdog setting and reading FLASH operation status, etc. Please refer to the corresponding description for the specific API and its corresponding functions.

The block of the embedded FLASH module consists of the following: Flash is divided into program flash (2MB) and data flash (128KB). Program flash is composed of two blocks, and a block is divided into 512 sectors, that is, the size of a sector is 2KB. Data flash is composed of a block, and a block is divided into 64 sectors, that is, the size of a sector is 2KB. Sector is the smallest unit of FLASH erase.

### 63.3.1 Read

The default state of the embedded flash memory is read. The main and NVR address space can be read-only in the read state. Users can directly use the address to read FLASH, but array reads attempted to invalid locations result in indeterminate data. Invalid locations occur when addressing is done to block that does not exist based on the number of blocks in that configuration (out-of-range addresses).

### 63.3.2 Program

A flash memory program sequence operates on any page within the flash core. In FC7240 products, in a single programming operation, up to 16 pages can be operated. One page is 2 words, and page is the smallest unit of single programming. Whenever the array is programmed, the ECC bits also get programmed. ECC is handled on a 64-bit boundary. In FC7240, one programming can be up to 32 words (that is, with 32 words as the boundary), and the minimum recommended is 2 words. When only 2 words are programmed, the value of the rest will remain unchanged. In addition, FC7240 supports whether to program the full F value into FLASH, please see the usage of the specific API

### 63.3.3 Erase

An erase operation changes the value stored in all bits of the selected sector or block to logic 1. The erase sequence is fully automated within the flash memory, please see the usage of the specific API.

### 63.3.4 Array Integrity Self Check

Array integrity is checked using a predefined address sequence (Can be selected by the enumeration FLASH_API_ARRAY_SEQ_TYPE). Users generally program their own application code into FLASH, specify the address of the program code, and select the appropriate check sequence to calculate, so as to obtain a unique result.

### 63.3.5 Margin Read Self Check

Margin read check uses the Array integrity check interface to self-check. Its operation is basically the same as an array integrity check. The user selects the margin read level (erased bits or programmed bits), specifies the MISR seed value, selects the check flash address, and finally selects the corresponding sequence to calculate the result. Compare the results obtained with the results generated by the tool to know the status of the hardware.

### 63.3.6 ECC Logic Check

The ECC logic check function is used to check whether the ECC logic inside FLASH is correct. Since the ECC in FLASH is processed on the 64-bit boundary, the user needs to provide an address, a 64-bit data and its expected ECC value to self-check ECC logic, and the hardware will recalculate according to the provided data ECC. After the user reads the address again, check the status of FLASH (single-bit error or multi-bit error) to verify the function of ECC logic.

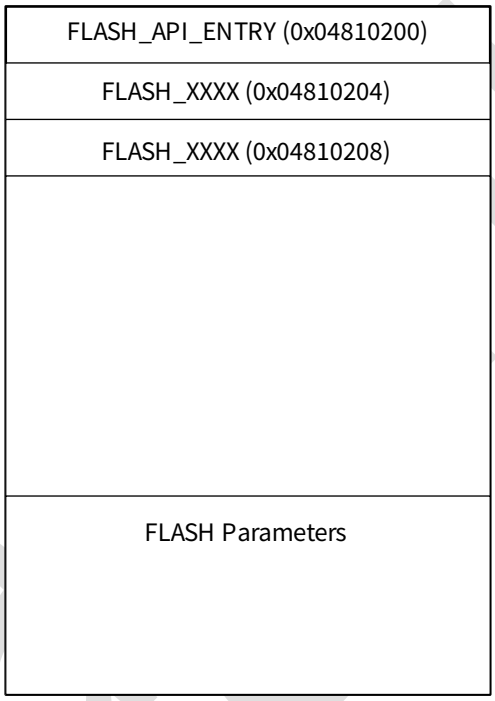
### 63.3.7 EDC after ECC Logic Check

EDC is the function of the hardware to check the data again according to a specific algorithm after completing the ECC check of the data. To verify the correct function of the EDC hardware, we provide a self-check on the EDC. The user needs to provide an address, a 64-bit data and its expected EDC value to self-check EDC logic, and the hardware will recalculate according to the provided data EDC. After the user reads the address again, check the status of FLASH (EDC error flag) to verify the function of EDC logic.

## 63.4 FLASH API Function

Flash API contains several parts, which are all implemented on ROM, and the start address is 0x04810200. The first part contains all API function entry addresses. The second part contains flash parameters. For details of flash parameters, please contact us.

**Figure 151. Flash API struct**




The following table lists all the functions of the FLASH API.

| Number | Function Name | Offset | Description |
|---|---|---|---|
| 0 | FlashRomApiVersion | 0x0 | Major/Minor version of the flash API |
| 1 | FLASH_DRV_EraseBlock | 0x4 | Execute the operation of erasing a flash block |
| 2 | FLASH_DRV_EraseBlock_Clear | 0x8 | Clean flash after erasing a flash block is completed |
| 3 | FLASH_DRV_EraseSector | 0xC | Execute the operation of erasing a flash sector |
| 4 | FLASH_DRV_EraseSector_Clear | 0x10 | Clean flash after erasing a flash sector is completed |
| 5 | FLASH_DRV_EraseNvr | 0x14 | Execute the operation of erasing NVR in flash |
| 6 | FLASH_DRV_EraseNvr_Clear | 0x18 | Clean flash after erasing NVR in flash is completed |
| 7 | FLASH_DRV_Program | 0x1C | Execute the operation of programming flash |
| 8 | FLASH_DRV_Program_Clear | 0x20 | Clean flash after programming flash is completed |

| | | | |
|---|---|---|---|
| 9 | FLASH_DRV_Program_Nvr | 0x24 | Execute the operation of programming NVR |
| 10 | FLASH_DRV_Program_Nvr_Clear | 0x28 | Clean flash after programming NVR is completed |
| 11 | FLASH_DRV_Array_Integrity_Check | 0x2C | Execute array integrity check of flash in normal mode |
| 12 | FLASH_DRV_Margin_Read_Check | 0x30 | Execute margin read check of flash in normal mode |
| 13 | FLASH_DRV_Read_Status_Check | 0x34 | Check the status of flash in normal mode |
| 14 | FLASH_DRV_Read_Status_Check_Clear | 0x38 | Clear the status flag of flash in normal mode |
| 15 | FLASH_DRV_HV_Status_Check | 0x3C | Check whether the flash is in the state of high voltage operation |
| 16 | FLASH_DRV_Test_Status_Check | 0x40 | Check whether array integrity or margin read is completed and return data |
| 17 | FLASH_DRV_Wdog_Configure | 0x44 | Configure watchdog in flash |
| 18 | FLASH_DRV_Wdog_Status | 0x48 | Check whether the watchdog is timed out |
| 19 | FLASH_DRV_Wdog_Status_Clear | 0x4C | Clear the status flag of the watchdog (Generally not required) |
| 20 | FLASH_DRV_EDC_Check | 0x50 | Execute EDC check of flash |
| 21 | FLASH_DRV_ECC_Check | 0x54 | Execute ECC check of flash |
| 22 | FLASH_DRV_Test_Mode_Close | 0x58 | Disable Test mode |
| 23 | FLASH_DRV_Read_Wait_CFG | 0x5C | Configure the read wait number of flash |
| 24 | FLASH_DRV_ENABLE_HOLD_CFG | 0x60 | Configure the hold function of flash |
| 25 | FLASH_DRV_READ_ADDR_FUNC | 0x64 | Read the first failing address in related events |
| 26 | FLASH_DRV_Test_Cfg_FUNC | 0x68 | Configure ECC/EDC/SBCE enable bit |
| 27 | FLASH_DRV_Array_Integrity_Suspend_FUNC | 0x6C | Configure flash array integrity suspend function |
| 28 | FLASH_DRV_Test_Mode_Open | 0x70 | Flash test mode open |
| 29 | FLASH_DRV_Sector_Program | 0x74 | Perform flash sector programming operations |
| 30 | FLASH_DRV_Read_Status_Check_128 | 0x78 | Read the status information of flash in parallel mode |
| 31 | FLASH_DRV_Read_Status_Check_Clear_128 | 0x7C | Clear the status information of flash in parallel mode |
| 32 | FLASH_DRV_Array_Integrity_Check_128 | 0x80 | Execute array integrity check of flash in parallel mode |
| 33 | FLASH_DRV_Margin_Read_Check_128 | 0x84 | Execute margin read check of flash in parallel mode |
| 34 | FLASH_DRV_CTL_CFG_FUNC | 0x88 | Configure some function bit(safer/pgff etc) |
| 35 | FLASH_DRV_READ_MASTER_ID_FUNC | 0x8C | Read master ID |
| 36 | FLASH_DRV_EDC_Check_128 | 0x90 | Execute EDC check of flash in parallel mode |
| 37 | FLASH_DRV_ECC_Check_128 | 0x94 | Execute ECC check of flash in parallel mode |
| 38 | FLASH_DRV_EAC_CHECK_128_HI | 0x98 | Execute EAC check of flash in parallel mode high |
| 39 | FLASH_DRV_AEC_CHECK | 0x9C | Execute Address injection check of flash in normal mode |
| 40 | FLASH_DRV_MCTL_CFG_FUNC | 0xA0 | Configure some function bit |

Flagchip

| | | | |
|---|---|---|---|
| 41 | FLASH_DRV_AMCTL_CFG_FUNC | 0xA4 | Configure some function bit |
| 42 | FLASH_DRV_READ_PE_MASTER_ID_FUNC | 0xA8 | Read master ID (which operate flash) |
| 43 | FLASH_DRV_READ_PE_AMASTER_ID_FUNC | 0xAC | Read master ID (which operate flash) |
| 44 | FLASH_DRV_EAC_CHECK_128_LO | 0xB0 | Execute EAC check of flash in parallel mode low |
| 45 | FLASH_DRV_EDCA_CHECK_128_HI | 0xB4 | Execute EDC address injection check of flash in parallel mode high |
| 46 | FLASH_DRV_EDCA_CHECK_128_LO | 0xB8 | Execute EDC address injection check of flash in parallel mode low |
| 47 | FLASH_DRV_ABORT_PE | 0xBC | Execute flash abort operation |
| 48 | FLASH_DRV_WRITE_ADR_FUNC | 0xC0 | Write ADR information |

### 63.4.1 FlashRomApiVersion

Offset: 0h

#### *63.4.1.1 Function Header*

***uint32_t FlashRomApiVersion (void)***

#### *63.4.1.2 Description*

| | |
|---|---|
| **Function name** | FlashRomApiVersion |
| **Function prototype** | uint32_t FlashRomApiVersion (void) |
| **Function description** | View FLASH API Major/Minor version number |
| **Input parameter** | Nothing |
| **Output parameter** | Nothing |
| **Return value** | Returns the FLASH API Major/Minor version number |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.1.3 Returns*

Returns the FLASH API Major version number is a 32-bit hexadecimal value

### 63.4.2 FLASH_DRV_EraseBlock

Offset: 4h

#### *63.4.2.1 Function Header*

***status_t FLASH_DRV_EraseBlock(FLASH_API_BLOCK_SELECT_TYPE blk_se, FLASH_API_ENABLE_TYPE int_en, FLASH_API_SIZE_TYPE type)***

#### *63.4.2.2 Description*

| | |
|---|---|
| **Function name** | **FLASH_ DRV_ EraseBlock** |
| **Function prototype** | status_t FLASH_DRV_EraseBlock(FLASH_API_BLOCK_SELECT_TYPE blk_sel, FLASH_API_ENABLE_TYPE int_en, FLASH_API_SIZE_TYPE type) |
| **Function description** | Execute erase the contents of block0/block1/block2/block3 in flash command |
| **Input parameter** | blk_sel: Select the number of the block<br>See section: FLASH_ API_BLOCK_ SELECT_ TYPE to view more allowable value ranges of this parameter |

| | |
|---|---|
| | int_en: Configure whether to enable flash interrupt<br>See section: FLASH_ API_ENABLE_ TYPE to view more allowable value ranges of this parameter<br>type: Choose the product type, 2M FLASH<br>See section: FLASH_ API_SIZE_ TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The status of the erase block command<br>***Note:*** After the function is executed, flash does not finish erasing, and a flash interrupt will be triggered after completion, when the flash interrupt is enabled. The user needs to wait until it is finished before operating the flash. |
| **Precondition** | Unlock flash |
| **Called function** | Nothing |

#### *63.4.2.3 Arguments*

FLASH_ API_BLOCK_ SELECT_ TYPE is used to select which block in flash to erase.

| FLASH_API_BLOCK_SELECT_TYPE | Description |
|---|---|
| FLASH_BLOCK_SELECT0 | Program Flash Block0 |
| FLASH_BLOCK_SELECT1 | Program Flash Block1 |
| FLASH_DATA_BLOCK_SELECT0 | Data Flash Block0 |

FLASH_ API_ENABLE_TYPE is used to select Whether to enable flash interrupt.

| FLASH_API_ENABLE_TYPE | Description |
|---|---|
| FLASH_API_DISABLE | Flash interrupt disable |
| FLASH_API_ENABLE | Flash interrupt enable |

FLASH_ API_SIZE_TYPE is used to select which product.

| FLASH_API_SIZE_TYPE | Description |
|---|---|
| FLASH_API_SIZE_2M | 2M Flash |

#### *63.4.2.4 Returns*

status_t is used to know the result of function execution.

| status_t | Description |
|---|---|
| STATUS_SUCCESS | Success (0x1) |
| STATUS_ERROR | Error (0x2) |
| STATUS_PARA_ERROR | Parameter error (0x3) |
| STATUS_ECC_ERROR | In the previous read flash data, double bit detection occurred (0x11) |
| STATUS_SBC_ERROR | In the previous read flash data, single bit correction occurred (0x21) |
| STATUS_AEE_ERROR | In the previous read flash data, address encode error occurred (0x41) |
| STATUS_EDC_ERROR | In the previous read flash data, an EDC after ECC error occurred (0x81) |

| | |
|---|---|
| STATUS_SPACE_ERROR | In the previous read flash data, a Space error occurred (0x101) |
| STATUS_TWO_ERROR | In the previous read flash data, a Two error occurred (0x201) |
| STATUS_ONE_ERROR | In the previous read flash data, a One error occurred (0x401) |
| STATUS_BLANK_ERROR | In the previous read flash data, a Blank error occurred (0x801) |
| STATUS_RWW_ERROR | In the previous read flash data, a read-while-write error occurred (0x1001) |
| STATUS_PEP_ERROR | Flash is not unlocked during the program or erase operation (0x2001) |
| STATUS_PES_ERROR | Program and Erase Sequence error (0x4001) |
| STATUS_HVOP | Flash is in high voltage operation (0x8001) |
| STATUS_PEG_FAILED | Flash program or erase is failed (0x10001) |
| STATUS_WDG_ERROR | Watchdog is timed out (0x20001) |
| STATUS_NVR_TIMEOUT_ERROR | Read NVR Timeout error (0x40001) |
| STATUS_RE_ERROR | Reset error encountered (0x80001) |
| STATUS_NVR_LOCK_ERROR | NVR space lock status (0x100001) |
| STATUS_ARRAY_BP | Flash array break point error occurred (0x200001) |
| STATUS_ENC_ERROR | Flash Encode error (0x400001) |
| STATUS_NVR_CFG_LD_EERROR | NVR CFG load error(0x800001) |
| STATUS_NVR_LD_ERROR | NVR load error(0x1000001) |
| STATUS_OTA_LD_ERROR | OTA load error(0x2000001) |
| STATUS_ANA_ERROR | Flash analog error(0x4000001) |
| STATUS_STRICT_PEG_FAILED | Flash strict program/erase failed(0x8000001) |
| STATUS_LC_SCAN_ERROR | Flash scan error(0x10000001) |
| FLASH_HSM_LD_ERROR | HSM load error(0x20000001) |
| FLASH_NVR_KEY_ERROR | NVR Key error(0x40000001) |

#### *63.4.2.5 Example*

```
status_t status;
FLASH_ API_BLOCK_ SELECT_ TYPE blk_sel = FLASH_BLOCK_SELECT0;
FLASH_API_ENABLE_TYPE int_en = FLASH_API_ENABLE;
FLASH_API_SIZE_TYPE type = FLASH_API_SIZE_2M;
status=FLASH_DRV_EraseBlock(blk_sel,int_en,type);  // Execute flash erase block command. After completion, a flash interrupt will be triggered
```

***Note:*** Register the flash interrupt function (use FLASH_DRV_EraseBlock_Clear in the interrupt function after successfully erasing the Flash)

### 63.4.3 FLASH_DRV_EraseBlock_Clear

Offset : 8h

#### *63.4.3.1 Function Header*

***status_t FLASH_DRV_EraseBlock_Clear(void)***

#### *63.4.3.2 Description*

| | |
|---|---|
| **Function name** | **FLASH_ DRV_ EraseBlock_Clear** |
| **Function prototype** | status_t FLASH_DRV_EraseBlock_Clear(void) |
| **Function description** | Clean flash after erasing a flash block is completed |

| | |
|---|---|
| | ***Note:*** if flash interrupt is enabled, this function needs to be called in Flash ISR. |
| **Input parameter** | Nothing |
| **Output parameter** | Nothing |
| **Return value** | The result of the erase flash block |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.3.3 Arguments*

None

#### *63.4.3.4 Returns*

Please see the definition of status_t.

### 63.4.4 FLASH_DRV_EraseSector

Offset : Ch

#### *63.4.4.1 Function Header*

```
typedef struct {
FLASH_API_BLOCK_SELECT_TYPE blk_sel;
uint32_t dest;
}FLASH_API_ERASESECTOR_TYPE;

status_t FLASH_DRV_EraseSector(FLASH_DRV_ERASESECTOR_CFG_T * flash_api_cfg, FLASH_API_ENABLE_TYPE
int_en, FLASH_API_SIZE_TYPE type)
```

#### *63.4.4.2 Description*

| | |
|---|---|
| **Function name** | **FLASH_ DRV_ EraseSector** |
| **Function prototype** | status_t FLASH_DRV_EraseSector(FLASH_DRV_ERASESECTOR_CFG_T * flash_api_cfg, FLASH_API_ENABLE_TYPE int_en, FLASH_API_SIZE_TYPE type) |
| **Function description** | Execute the operation of erasing a flash sector |
| **Input parameter** | See section: FLASH_API_ERASESECTOR_TYPE to view more allowable value ranges of this parameter.<br>int_en: Configure whether to enable flash interrupt<br>See section: FLASH_ API_ENABLE_ TYPE to view more allowable value ranges of this parameter<br>type: Choose the product type, 2M FLASH<br>See section: FLASH_ API_SIZE_ TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of the erase sector command<br>***Note:*** After the function is executed, flash does not finish erasing, and a flash interrupt will be triggered after completion, if the flash interrupt is enabled. The user needs to wait until it is finished before operating the flash. |
| **Precondition** | Unlock flash |
| **Called function** | Nothing |

#### 63.4.4.3 Arguments

FLASH_ API_ ERASESECTOR_ TYPE is used to pass in the address and block number of the erasing sector.

| FLASH_API_ERASESECTOR_TYPE | Description |
|---|---|
| FLASH_API_BLOCK_SELECT_TYPE blk_sel | The block number of the erase sector. See FLASH_API_BLOCK_SELECT_TYPE |
| uint32_t dest | Sector address |

#### 63.4.4.4 Returns

Please see the definition of status_t.

#### 63.4.4.5 Example

```
status_t status;
FLASH_ API_BLOCK_ SELECT_ TYPE blk_sel = FLASH_BLOCK_SELECT0;
uint32_t dest = 0x0;
FLASH_API_ERASESECTOR_TYPE flash_api_cfg;
flash_api_cfg.blk_sel = blk_sel;
flash_api_cfg.dest = dest;
FLASH_API_ENABLE_TYPE int_en = FLASH_API_ENABLE;
FLASH_API_SIZE_TYPE type = FLASH_API_SIZE_2M;
status=FLASH_DRV_EraseSector(&flash_api_cfg, int_en, type); // Execute flash erase sector command. After completion, a flash interrupt will be triggered
```

### 63.4.5 FLASH_DRV_EraseSector_Clear

Offset: 10h

#### 63.4.5.1 Function Header

***status_t FLASH_DRV_EraseSector_Clear(void)***

#### 63.4.5.2 Description

| Function name | FLASH_ DRV_ EraseSector_Clear |
|---|---|
| Function prototype | status_t FLASH_DRV_EraseSector_Clear(void) |
| Function description | Clean flash after erasing a flash sector is completed<br>***Note:*** If the flash interrupt is enabled, this function needs to be called in Flash ISR. |
| Input parameter | See section: FLASH_ API_ENABLE_ TYPE to view more allowable value ranges of this parameter |
| Output parameter | Nothing |
| Return value | The result of the erase flash sector |
| Precondition | Nothing |
| Called function | Nothing |

#### 63.4.5.3 Arguments

None

#### 63.4.5.4 Returns

Please see the definition of status_t.

### 63.4.6 FLASH_DRV_EraseNvr

Offset: 14h

#### 63.4.6.1 Function Header

*status_t FLASH_DRV_EraseNvr(uint32_t addr, FLASH_API_ENABLE_TYPE int_en)*

#### 63.4.6.2 Description

| **Function name** | **FLASH_ DRV_ EraseNvr** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_EraseNvr(uint32_t addr, FLASH_API_ENABLE_TYPE int_en) |
| **Function description** | Execute the operation of erasing flash NVR |
| **Input parameter** | addr : nvr sector address<br>int_en: Configure whether to enable flash interrupt<br>See section: FLASH_ API_ENABLE_ TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | Result of executing erase flash NVR command<br>***Note:*** After the function is executed, flash does not finish erasing, and a flash interrupt will be triggered after completion when the flash interrupt is enabled. The user needs to wait until it is finished before operating the flash. |
| **Precondition** | Unlock flash |
| **Called function** | Nothing |

#### 63.4.6.3 Arguments

Please see the definition of FLASH_API_ENABLE_TYPE.

#### 63.4.6.4 Returns

Please see the definition of status_t.

### 63.4.7 FLASH_DRV_EraseNvr_Clear

Offset: 18h

#### 63.4.7.1 Function Header

*status_t FLASH_DRV_EraseNvr_Clear(void)*

#### 63.4.7.2 Description

| **Function name** | **FLASH_ DRV_ EraseNvr_Clear** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_EraseNvr_Clear(void) |
| **Function description** | Clean flash after erasing flash NVR is completed<br>***Note:*** This function needs to be called in Flash ISR, when the flash interrupt is enabled. |
| **Input parameter** | Nothing |
| **Output parameter** | Nothing |

| | |
|---|---|
| **Return value** | The result of the erase flash NVR |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.7.3 Arguments*

None

#### *63.4.7.4 Returns*

Please see the definition of status_t.

### 63.4.8 FLASH_DRV_Program

Offset: 1Ch

#### *63.4.8.1 Function Header*

```
typedef struct {
uint32_t dest;
FLASH_API_PRGM_SIZE_TYPE size;
uint32_t *pData;
FLASH_API_PRGM_WDG_TUNE_TYPE wdg_tune;
FLASH_API_REG_BIT_TYPE pgff
}FLASH_API_PRGM_CFG_TYPE;

status_t FLASH_DRV_Program(FLASH_API_PRGM_CFG_TYPE * flash_api_cfg, FLASH_API_ENABLE_TYPE int_en,
FLASH_API_SIZE_TYPE type)
```

#### *63.4.8.2 Description*

| | |
|---|---|
| **Function name** | **FLASH_ DRV_ Program** |
| **Function prototype** | status_t FLASH_DRV_Program(FLASH_API_PRGM_CFG_TYPE * flash_api_cfg, FLASH_API_ENABLE_TYPE int_en , FLASH_API_SIZE_TYPE type) |
| **Function description** | Execute the operation of programming flash |
| **Input parameter** | See section: FLASH_API_PRGM_CFG_TYPE to view more allowable value ranges of this parameter<br>int_en: Configure whether to enable flash interrupt<br>See section: FLASH_ API_ENABLE_ TYPE to view more allowable value ranges of this parameter<br>type: Choose the product type, 2M FLASH<br>See section: FLASH_ API_SIZE_ TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | Result of executing programming flash command<br>***Note:*** After the function is executed, flash does not finish programing, and a flash interrupt will be triggered after completion, when the flash interrupt is enabled. The user needs to wait until it is finished before operating the flash. |
| **Precondition** | Unlock flash |
| **Called function** | Nothing |

#### 63.4.8.3 Arguments

FLASH_ API_ PRGM_ CFG_ TYPE is used to pass the parameters required for flash programming.

| FLASH_API_PRGM_CFG_TYPE | Description |
|---|---|
| uint32_t dest | Program address |
| FLASH_API_PRGM_SIZE_TYPE size | Programming size (minimum 8 bytes; maximum 128 bytes) |
| uint32_t *pData | Programming data |
| FLASH_API_PRGM_WDG_TUNE_TYPE wdg_tune | WDOG timeout config |
| FLASH_API_REG_BIT_TYPE pgff | pgff control bit |

| FLASH_API_PRGM_SIZE_TYPE | Description |
|---|---|
| FLASH_PRGM_SIZE1 | 8byte |
| FLASH_PRGM_SIZE2 | 16byte |
| FLASH_PRGM_SIZE3 | 24byte |
| FLASH_PRGM_SIZE4 | 32byte |
| FLASH_PRGM_SIZE5 | 40byte |
| FLASH_PRGM_SIZE6 | 48byte |
| FLASH_PRGM_SIZE7 | 56byte |
| FLASH_PRGM_SIZE8 | 64byte |
| FLASH_PRGM_SIZE9 | 72byte |
| FLASH_PRGM_SIZE10 | 80byte |
| FLASH_PRGM_SIZE11 | 88byte |
| FLASH_PRGM_SIZE12 | 96byte |
| FLASH_PRGM_SIZE13 | 104byte |
| FLASH_PRGM_SIZE14 | 112byte |
| FLASH_PRGM_SIZE15 | 120byte |
| FLASH_PRGM_SIZE16 | 128byte |

| FLASH_API_PRGM_WDG_TUNE_TYPE | Description |
|---|---|
| WDG_TUNE_DISABLE | Use the default watchdog timeout configuration |
| WDG_TUNE_ENABLE | Use the user's watchdog configuration |

| FLASH_API_REG_BIT_TYPE | Description |
|---|---|
| FLASH_REG_BIT_CFG_DISABLE | Disable reg bit |
| FLASH_REG_BIT_CFG_ENABLE | Enable reg bit |
| FLASH_REG_BIT_CFG_HOLD | Hold reg bit |

#### 63.4.8.4 Returns

Please see the definition of status_t.

#### 63.4.8.5 Example

```
status_t status;
FLASH_ API_PRGM_SIZE_TYPE size = FLASH_PRGM_SIZE2; //4 word prgm
FLASH_API_PRGM_WDG_TUNE_TYPE wdg_tune=WDG_TUNE_DISABLE;  //Recommended use
FLASH_API_REG_BIT_TYPE pgff = FLASH_REG_BIT_CFG_DISABLE; //Recommended use
FLASH_API_ENABLE_TYPE int_en = FLASH_API_ENABLE;
FLASH_API_SIZE_TYPE type = FLASH_API_SIZE_2M;
uint32_t data[4];
data[0]=0x12345678;data[1]=0x12345678;data[2]=0xA5;data[3]=0x5A;  //program data
uint32_t dest = 0x0;       //program start address
FLASH_API_PRGM_CFG_TYPE flash_api_cfg;
flash_api_cfg.dest = dest;
flash_api_cfg.size = size;
flash_api_cfg.pData = data;
flash_api_cfg.wdg_tune = wdg_tune;
flash_api_cfg.pgff = pgff;
status=FLASH_DRV_Program(&flash_api_cfg, int_en, type,); // Execute flash program command. After completion, a flash interrupt will be triggered. Need to call the corresponding clear function in the flash interrupt function to clear the flash state.
```

### 63.4.9 FLASH_DRV_Program_Clear

Offset: 20h

#### 63.4.9.1 Function Header

***status_t FLASH_DRV_Program_Clear(void)***

#### 63.4.9.2 Description

| | |
|---|---|
| **Function name** | **FLASH_DRV_Program_Clear** |
| **Function prototype** | status_t FLASH_DRV_Program_Clear(void) |
| **Function description** | Clean flash after programming flash is completed<br>***Note:*** This function needs to be called in Flash ISR, when the flash interrupt is enabled. |
| **Input parameter** | Nothing |
| **Output parameter** | Nothing |
| **Return value** | The result of the program flash |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### 63.4.9.3 Arguments

None

#### 63.4.9.4 Returns

Please see the definition of status_t.

### 63.4.10 FLASH_DRV_Program_Nvr

Offset: 24h

#### *63.4.10.1 Function Header*

*typedef struct {*
*uint32_t dest;*
*FLASH_API_PRGM_SIZE_TYPE size;*
*uint32_t *pData;*
*FLASH_API_PRGM_WDG_TUNE_TYPE wdg_tune;*
*FLASH_API_REG_BIT_TYPE pgff*
*}FLASH_API_PRGM_CFG_TYPE;*

*status_t FLASH_DRV_Program_Nvr(FLASH_API_PRGM_CFG_TYPE * flash_api_cfg, FLASH_API_ENABLE_TYPE int_en)*

#### *63.4.10.2 Description*

| | |
|---|---|
| **Function name** | **FLASH_DRV_Program_Nvr** |
| **Function prototype** | status_t FLASH_DRV_Program_Nvr(FLASH_API_PRGM_CFG_TYPE * flash_api_cfg, FLASH_API_ENABLE_TYPE int_en ) |
| **Function description** | Execute the operation of programming flash NVR |
| **Input parameter** | See section: FLASH_API_PRGM_CFG_TYPE to view more allowable value ranges of this parameter.<br>See section: FLASH_ API_ENABLE_ TYPE to view more allowable value ranges of this parameter. |
| **Output parameter** | Nothing |
| **Return value** | Result of executing program flash NVR command<br>***Note:*** After the function is executed, flash does not finish programming, and a flash interrupt will be triggered after completion, when the flash interrupt is enabled. The user needs to wait until it is finished before operating the flash. |
| **Precondition** | Unlock flash |
| **Called function** | Nothing |

#### *63.4.10.3 Arguments*

FLASH_ API_ PRGM_ CFG_ TYPE is used to pass the parameters required for flash programming.

| **FLASH_API_PRGM_CFG_TYPE** | **Description** |
|---|---|
| uint32_t dest | Program address |
| FLASH_API_PRGM_SIZE_TYPE size | Programming size (minimum 8 bytes; maximum 128 bytes) |
| uint32_t *pData | Programming data |
| FLASH_API_PRGM_WDG_TUNE_TYPE wdg_tune | WDOG timeout config |
| FLASH_API_REG_BIT_TYPE pgff | pgff control bit |

| **FLASH_API_PRGM_WDG_TUNE_TYPE** | **Description** |
|---|---|
| WDG_TUNE_DISABLE | Use the default watchdog timeout configuration |
| WDG_TUNE_ENABLE | Use the user's watchdog configuration |

| FLASH_API_REG_BIT_TYPE | Description |
|---|---|
| FLASH_REG_BIT_CFG_DISABLE | Disable reg bit |
| FLASH_REG_BIT_CFG_ENABLE | Enable reg bit |
| FLASH_REG_BIT_CFG_HOLD | Hold reg bit |

#### *63.4.10.4 Returns*

Please see the definition of status_t.

### 63.4.11 FLASH_DRV_Program_Nvr_Clear

Offset: 28h

#### *63.4.11.1 Function Header*

***status_t FLASH_DRV_Program_Nvr_Clear(void)***

#### *63.4.11.2 Description*

| Function name | FLASH_DRV_Program_Nvr_Clear |
|---|---|
| Function prototype | status_t FLASH_DRV_Program_Nvr_Clear(void) |
| Function description | Clean flash after programming flash NVR is completed<br>***Note:*** This function needs to be called in Flash ISR, When the flash interrupt is enabled. |
| Input parameter | Nothing |
| Output parameter | Nothing |
| Return value | The result of the program flash NVR |
| Precondition | Nothing |
| Called function | Nothing |

#### *63.4.11.3 Arguments*

None

#### *63.4.11.4 Returns*

Please see the definition of status_t.

### 63.4.12 FLASH_DRV_Array_Integrity_Check

Offset: 2Ch

#### *63.4.12.1 Function Header*

```
typedef struct {
 uint32_t dest;
 uint32_t sed0;
 uint32_t sed1;
 uint32_t sed2;
 uint32_t sed3;
 uint32_t sed4;
```

*uint32_t sed5;*

*FLASH_API_WAIT_STATE_TYPE wait_number;*

*FLASH_API_MARGIN_LEVEL_TYPE mrv;*

*FLASH_API_ARRAY_INTEGRITY_SEQ_TYPE als;*

*FLASH_API_REG_BIT_TYPE bpe;*

*}FLASH_API_TEST_CHECK_CFG_TYPE;*

*status_t FLASH_DRV_Array_Integrity_Check(FLASH_API_TEST_CHECK_CFG_TYPE * flash_api_cfg)*

#### *63.4.12.2 Description*

| **Function name** | **FLASH_DRV_Array_Integrity_Check** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_Array_Integrity_Check(FLASH_API_TEST_CHECK_CFG_TYPE * flash_api_cfg) |
| **Function description** | Check flash array |
| **Input parameter** | See section: FLASH_API_TEST_CHECK_CFG_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing the array integrity command |
| **Precondition** | FLASH TEST MODE OPEN |
| **Called function** | Nothing |

#### *63.4.12.3 Arguments*

FLASH_API_TEST_CHECK_CFG_TYPE is used to pass the parameters required for the flash array integrity check.

| **FLASH_API_TEST_CHECK_CFG_TYPE** | **Description** |
|---|---|
| uint32_t dest | Array integrity check address |
| uint32_t sed0 | Seed value |
| uint32_t sed1 | Seed value |
| uint32_t sed2 | Seed value |
| uint32_t sed3 | Seed value |
| uint32_t sed4 | Seed value |
| uint32_t sed5 | Seed value |
| FLASH_API_WAIT_STATE_TYPE wait_number | The number of wait states |
| FLASH_API_MARGIN_LEVEL_TYPE mrv | Margin read value (not required) |
| FLASH_ARRAY_INTEGRITY_SEQ_TYPE als | Select the sequence of array integrity check |
| FLASH_API_REG_BIT_TYPE bpe | Enable breakpoint function |

| **FLASH_API_WAIT_STATE_TYPE** | **Description** |
|---|---|
| FLASH_WAIT_SELECT1 | Wait for 1 cycle when accessing the flash array |
| FLASH_WAIT_SELECT2 | Wait for 2 cycles when accessing the flash array |
| FLASH_WAIT_SELECT3 | Wait for 3 cycles when accessing the flash array |
| FLASH_WAIT_SELECT4 | Wait for 4 cycles when accessing the flash array |
| FLASH_WAIT_SELECT5 | Wait for 5 cycles when accessing the flash array |

| | |
|---|---|
| FLASH_WAIT_SELECT6 | Wait for 6 cycles when accessing the flash array |
| FLASH_WAIT_SELECT7 | Wait for 7 cycles when accessing the flash array |
| FLASH_WAIT_SELECT8 | Wait for 8 cycles when accessing the flash array |
| FLASH_WAIT_HOLD | Keep config unchanged |

| **FLASH_API_MARGIN_LEVEL_TYPE** | **Description** |
|---|---|
| FLASH_MARGIN_LEVEL0 | Margin level 0 (programmed level) |
| FLASH_MARGIN_LEVEL1 | Margin level 1 (erased level) |

| **FLASH_API_ARRAY_SEQ_TYPE** | **Description** |
|---|---|
| FLASH_ARRAY_SEQ1 | Select proprietary sequence |
| FLASH_ARRAY_SEQ2 | Select sequential sequence |

| **FLASH_API_REG_BIT_TYPE** | **Description** |
|---|---|
| FLASH_REG_BIT_CFG_DISABLE | Disable reg bit |
| FLASH_REG_BIT_CFG_ENABLE | Enable reg bit |
| FLASH_REG_BIT_CFG_HOLD | Hold reg bit |

#### *63.4.12.4 Returns*

Please see the definition of status_t.

#### *63.4.12.5 Example*

```
status_t status;
FLASH_ API_WAIT_STATE_TYPE wait_number = FLASH_WAIT_SELECT6; //Recommended use
FLASH_API_MARGIN_LEVEL_TYPE mrv=FLASH_MARGIN_LEVEL0;  //Recommended use
FLASH_API_ARRAY_SEQ_TYPE als=FLASH_ARRAY_SEQ1;   //Recommended use
FLASH_API_TEST_CHECK_CFG_TYPE flash_api_cfg;
flash_api_cfg.dest = 0x00000000;
flash_api_cfg.sed0 = 0x00000000; flash_api_cfg.sed1 = 0x00000000;flash_api_cfg.sed2 = 0x00000000;
flash_api_cfg.wait_number= wait_number;
flash_api_cfg.mrv = mrv;
flash_api_cfg.als = als;
flash_api_cfg.bpe = 0x0;
status=FLASH_DRV_Array_Integrity_Check(&flash_api_cfg);  //To execute the array integrity check command, you need to use the FLASH_DRV_Test_Status_Check function to obtain the calculation result. It may take some time to get the results.
```

### 63.4.13 FLASH_DRV_Margin_Read_Check

Offset: 30h

#### *63.4.13.1 Function Header*

```
typedef struct {
uint32_t dest;
```

*uint32_t sed0;*

*uint32_t sed1;*

*uint32_t sed2;*

*uint32_t sed3;*

*uint32_t sed4;*

*uint32_t sed5;*

*FLASH_API_WAIT_STATE_TYPE wait_number;*

*FLASH_API_MARGIN_LEVEL_TYPE mrv;*

*FLASH_API_ARRAY_INTEGRITY_SEQ_TYPE als;*

*FLASH_API_REG_BIT_TYPE bpe;*

*}FLASH_API_TEST_CHECK_CFG_TYPE;*

*status_t FLASH_DRV_Margin_Read_Check(FLASH_API_TEST_CHECK_CFG_TYPE * flash_api_cfg)*

#### *63.4.13.2 Description*

| **Function name** | **FLASH_DRV_Margin_Read_Check** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_Margin_Read_Check(FLASH_API_TEST_CHECK_CFG_TYPE * flash_api_cfg) |
| **Function description** | Check margin read |
| **Input parameter** | See section: FLASH_API_TEST_CHECK_CFG_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing the margin read command |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.13.3 Arguments*

FLASH_API_TEST_CHECK_CFG_TYPE is used to pass the parameters required for the flash margin read check.

| **FLASH_API_TEST_CHECK_CFG_TYPE** | **Description** |
|---|---|
| uint32_t dest | Array integrity check address |
| uint32_t sed0 | Seed value |
| uint32_t sed1 | Seed value |
| uint32_t sed2 | Seed value |
| uint32_t sed3 | Seed value |
| uint32_t sed4 | Seed value |
| uint32_t sed5 | Seed value |
| FLASH_API_WAIT_STATE_TYPE wait_number | The number of wait states |
| FLASH_API_MARGIN_LEVEL_TYPE mrv | Margin read value (not required) |
| FLASH_ARRAY_INTEGRITY_SEQ_TYPE als | Select the sequence of array integrity check |
| FLASH_API_REG_BIT_TYPE bpe | Enable breakpoint function |

| **FLASH_API_WAIT_STATE_TYPE** | **Description** |
|---|---|

| | |
|---|---|
| FLASH_WAIT_SELECT1 | Wait for 1 cycle when accessing the flash array |
| FLASH_WAIT_SELECT2 | Wait for 2 cycles when accessing the flash array |
| FLASH_WAIT_SELECT3 | Wait for 3 cycles when accessing the flash array |
| FLASH_WAIT_SELECT4 | Wait for 4 cycles when accessing the flash array |
| FLASH_WAIT_SELECT5 | Wait for 5 cycles when accessing the flash array |
| FLASH_WAIT_SELECT6 | Wait for 6 cycles when accessing the flash array |
| FLASH_WAIT_SELECT7 | Wait for 7 cycles when accessing the flash array |
| FLASH_WAIT_SELECT8 | Wait for 8 cycles when accessing the flash array |
| FLASH_WAIT_HOLD | Keep config unchanged |

| **FLASH_API_MARGIN_LEVEL_TYPE** | **Description** |
|---|---|
| FLASH_MARGIN_LEVEL0 | Margin level 0 (programmed level) |
| FLASH_MARGIN_LEVEL1 | Margin level 1 (erased level) |

| **FLASH_API_ARRAY_SEQ_TYPE** | **Description** |
|---|---|
| FLASH_ARRAY_SEQ1 | Select proprietary sequence |
| FLASH_ARRAY_SEQ2 | Select sequential sequence |

| **FLASH_API_REG_BIT_TYPE** | **Description** |
|---|---|
| FLASH_REG_BIT_CFG_DISABLE | Disable reg bit |
| FLASH_REG_BIT_CFG_ENABLE | Enable reg bit |
| FLASH_REG_BIT_CFG_HOLD | Hold reg bit |

#### *63.4.13.4 Returns*

Please see the definition of status_t.

#### *63.4.13.5 Example*

```
status_t status;
FLASH_ API_WAIT_STATE_TYPE wait_number = FLASH_WAIT_SELECT6; //Recommended use
FLASH_API_MARGIN_LEVEL_TYPE mrv=FLASH_MARGIN_LEVEL0;  //Recommended use
FLASH_API_ARRAY_SEQ_TYPE als=FLASH_ARRAY_SEQ1;  //Recommended use
FLASH_API_TEST_CHECK_CFG_TYPE flash_api_cfg;
flash_api_cfg.dest = 0x00000000;
flash_api_cfg.sed0 = 0x00000000; flash_api_cfg.sed1 = 0x00000000;flash_api_cfg.sed2 = 0x00000000;
flash_api_cfg.wait_number= wait_number;
flash_api_cfg.mrv = mrv;
flash_api_cfg.als = als;
flash_api_cfg.bpe = 0x0;
status=FLASH_DRV_Array_Integrity_Check(&flash_api_cfg);  //To execute the array integrity check command, you need to use the FLASH_DRV_Test_Status_Check function to obtain the calculation result. It may take some time to get the results.
```

### 63.4.14 FLASH_DRV_Read_Status_Check

Offset : 34h

#### 63.4.14.1 Function Header

*uint32_t FLASH_DRV_Read_Status_Check(FLASH_API_READ_STATUS_CHECK_TYPE flash_api_cfg)*

#### 63.4.14.2 Description

| | |
|---|---|
| **Function name** | **FLASH_DRV_Read_Status_Check** |
| **Function prototype** | uint32_t FLASH_DRV_Read_Status_Check(FLASH_API_READ_STATUS_CHECK_TYPE flash_api_cfg) |
| **Function description** | Check the status of pflash in normal mode |
| **Input parameter** | See section: FLASH_API_READ_STATUS_CHECK_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of the flash status |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### 63.4.14.3 Arguments

FLASH_API_READ_STATUS_CHECK_TYPE is used to pass the parameters.

| FLASH_API_READ_STATUS_CHECK_TYPE | Description |
|---|---|
| FLASH_ECC_CHECK_MASK | Check double bit detection occurred |
| FLASH_SBC_CHECK_MASK | Check single bit correction occurred |
| FLASH_AEE_CHECK_MASK | Check read-while-write error occurred |
| FLASH_EDC_CHECK_MASK | Check EDC after an ECC error occurred |
| FLASH_SPACE_CHECK_MASK | Check Space status |
| FLASH_TWO_CHECK_MASK | Check two status |
| FLASH_ONE_CHECK_MASK | Check one status |
| FLASH_BLANK_CHECK_MASK | Check Blank status |
| FLASH_RWW_CHECK_MASK | Check read-while-write error occurred |
| FLASH_PEP_CHECK_MASK | Check Program/Erase protection error occurred |
| FLASH_PES_CHECK_MASK | Check Program or erase sequence error |
| FLASH_HVOP_CHECK_MASK | Check Flash in program or erase sequence |
| FLASH_PEG_CHECK_MASK | Check Program/Erase result |
| FLASH_WDG_CHECK_MASK | Check WDOG status |
| FLASH_NVR_TIMEOUT_CHECK_MASK | Check NVR Timeout error occurred |
| FLASH_RE_CHECK_MASK | Check Reset error occurred |
| FLASH_NVR_LOCK_CHECK_MASK | Check NVR lock Status |
| FLASH_ARRAY_BP_CHECK_MASK | Check read-while-write error occurred |
| STATUS_ENC_CHECK_MASK | Check Flash Encode error |
| STATUS_NVR_CFG_LD_CHECK_MASK | Check NVR CFG load error |
| STATUS_NVR_LD_CHECK_MASK | Check NVR load error |
| STATUS_OTA_LD_CHECK_MASK | Check OTA load error |
| STATUS_ANA_CHECK_MASK | Check Flash analog error |
| STATUS_STRICT_PEG_CHECK_MASK | Check Flash strict program/erase failed |
| STATUS_LC_SCAN_CHECK_MASK | Check Flash scan error |

| | |
|---|---|
| FLASH_HSM_LD_CHECK_MASK | Check HSM load error |
| FLASH_NVR_KEY_CHECK_MASK | Check NVR Key error |

#### *63.4.14.4 Returns*

Please see the definition of status_t.

### 63.4.15 FLASH_DRV_Read_Status_Check_Clear

Offset: 38h

#### *63.4.15.1 Function Header*

***status_t FLASH_DRV_Read_Status_Check_Clear(FLASH_API_READ_STATUS_CLEAR_TYPE flash_api_cfg)***

#### *63.4.15.2 Description*

| | |
|---|---|
| **Function name** | **FLASH_DRV_Read_Status_Check_Clear** |
| **Function prototype** | status_t FLASH_DRV_Read_Status_Check_Clear (FLASH_API_READ_STATUS_CLEAR_TYPE flash_api_cfg) |
| **Function description** | Clear flash status flag |
| **Input parameter** | See section: FLASH_API_READ_STATUS_CLEAR_TYPE to view more allowable value ranges of this parameter. |
| **Output parameter** | Nothing |
| **Return value** | The result of clearing flash status |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.15.3 Arguments*

FLASH_API_READ_STATUS_CLEAR_TYPE is used to pass the parameters.

| **FLASH_API_READ_STATUS_CLEAR_TYPE** | **Description** |
|---|---|
| FLASH_CLEAR_ECC_MASK | Clear double-bit error |
| FLASH_CLEAR_SBC_MASK | Clear single-bit error |
| FLASH_CLEAR_AEE_MASK | Clear AEE error |
| FLASH_CLEAR_EDC_MASK | Clear EDC error |
| FLASH_CLEAR_SPACE__MASK | Clear Space error |
| FLASH_CLEAR_TWO_MASK | Clear Two error |
| FLASH_CLEAR_ONE_MASK | Clear One error |
| FLASH_CLEAR_BLANK_MASK | Clear Blank error |
| FLASH_CLEAR_RWW_MASK | Clear RWW error |
| FLASH_CLEAR_PEP_MASK | Clear PEP error |
| FLASH_CLEAR_PES_MASk | Clear PES error |
| FLASH_CLEAR_NVR_TIMEOUT_MASK | Clear NVR timeout error |
| FLASH_CLEAR_ARRAY_BP_MASK | Clear array break point error |
| FLASH_CLEAR_ENC_MASK | Clear ENC error |

#### 63.4.15.4 Returns

Please see the definition of status_t.

### 63.4.16 FLASH_DRV_HV_Status_Check

Offset: 3Ch

#### 63.4.16.1 Function Header

*status_t FLASH_DRV_HV_Status_Check(void)*

#### 63.4.16.2 Description

| | |
|---|---|
| **Function name** | **FLASH_DRV_HV_Status_Check** |
| **Function prototype** | status_t FLASH_DRV_HV_Status_Check(void) |
| **Function description** | Check flash high voltage status |
| **Input parameter** | Nothing |
| **Output parameter** | Nothing |
| **Return value** | Flash high voltage status |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### 63.4.16.3 Arguments

None

#### 63.4.16.4 Returns

Please see the definition of status_t.

### 63.4.17 FLASH_DRV_Test_Status_Check

Offset: 40h

#### 63.4.17.1 Function Header

```
typedef struct {
uint32_t data0;
uint32_t data1;
uint32_t data2;
uint32_t data3;
uint32_t data4;
uint32_t data5;
}FLASH_API_TEST_CHECK_RESULT;
```

*status_t FLASH_DRV_Test_Status_Check (FLASH_API_TEST_CHECK_RESULT * flash_api_cfg)*

#### 63.4.17.2 Description

| | |
|---|---|
| **Function name** | **FLASH_DRV_Test_Status_Check** |
| **Function prototype** | status_t FLASH_DRV_Test_Status_Check (FLASH_API_TEST_CHECK_RESULT * flash_api_cfg) |
| **Function description** | After the array integrity check or margin read check, the calculation result is obtained through this function |

| | |
|---|---|
| **Input parameter** | See section: FLASH_API_TEST_CHECK_RESULT to view more allowable value ranges of this parameter |
| **Output parameter** | Test Check compute result |
| **Return value** | The result of Test status |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.17.3 Arguments*

FLASH_API_TEST_CHECK_RESULT is used to pass the parameters required for the flash test status check.

| FLASH_API_TEST_CHECK_RESULT | Description |
|---|---|
| uint32_t data0 | Return test check data0 |
| uint32_t data1 | Return test check data1 |
| uint32_t data2 | Return test check data2 |
| uint32_t data3 | Return test check data3 |
| uint32_t data4 | Return test check data4 |
| uint32_t data5 | Return test check data5 |

#### *63.4.17.4 Returns*

Please see the definition of status_t.

### 63.4.18 FLASH_DRV_Wdog_Configure

Offset: 44h

#### *63.4.18.1 Function Header*

```
typedef struct {
FLASH_API_WDG_TIMEOUT_TYPE time;
FLASH_API_REG_BIT_TYPE wint_en;
FLASH_API_REG_BIT_TYPE wen;
}FLASH_API_WDG_CFG_TYPE;

status_t FLASH_DRV_Wdog_Configure (FLASH_API_WDG_CFG_TYPE * flash_api_cfg)
```

#### *63.4.18.2 Description*

| | |
|---|---|
| **Function name** | **FLASH_DRV_Wdog_Configure** |
| **Function prototype** | status_t FLASH_DRV_Wdog_Configure (FLASH_API_WDG_CFG_TYPE * flash_api_cfg) |
| **Function description** | Configure Flash watchdog |
| **Input parameter** | See section: FLASH_API_WDG_CFG_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of config flash watchdog |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### 63.4.18.3 Arguments

FLASH_API_WDG_CFG_TYPE is used to pass the parameters required for flash watchdog.

| FLASH_API_WDG_CFG_TYPE | Description |
|---|---|
| FLASH_API_WDG_TIMEOUT_TYPE time | Watchdog timeout time |
| FLASH_API_REG_BIT_TYPE wint_en | Watchdog interrupt configure |
| FLASH_API_REG_BIT_TYPE wen | Watchdog configure (enable/disable) |

| FLASH_API_WDG_TIMEOUT_TYPE | Description |
|---|---|
| FLASH_WDG_TIMEOUT_SELECT0 | Timeout set to 10x (400 clocks default) |
| FLASH_WDG_TIMEOUT_SELECT1 | Timeout set to 5x |
| FLASH_WDG_TIMEOUT_SELECT2 | Timeout set to 100x |
| FLASH_WDG_TIMEOUT_SELECT3 | Timeout set to 1000x |
| FLASH_WDG_TIMEOUT_HOLD | Timeout configure uncharge |

| FLASH_API_REG_BIT_TYPE | Description |
|---|---|
| FLASH_REG_BIT_CFG_DISABLE | Disable reg bit |
| FLASH_REG_BIT_CFG_ENABLE | Enable reg bit |
| FLASH_REG_BIT_CFG_HOLD | Hold reg bit |

#### 63.4.18.4 Returns

Please see the definition of status_t.

### 63.4.19 FLASH_DRV_Wdog_Status

Offset: 48h

#### 63.4.19.1 Function Header

*status_t FLASH_DRV_Wdog_Status(void)*

#### 63.4.19.2 Description

| Function name | FLASH_DRV_Wdog_Status |
|---|---|
| Function prototype | status_t FLASH_DRV_Wdog_Status(void) |
| Function description | Check flash watchdog timeout status |
| Input parameter | Nothing |
| Output parameter | Nothing |
| Return value | The result of flash watchdog timeout status |
| Precondition | Nothing |
| Called function | Nothing |

#### 63.4.19.3 Arguments

None

#### 63.4.19.4 Returns

Please see the definition of status_t.

### 63.4.20 FLASH_DRV_Wdog_Status_Clear

Offset: 4Ch

#### 63.4.20.1 Function Header

*status_t FLASH_DRV_Wdog_Status_Clear(void)*

#### 63.4.20.2 Description

| **Function name** | **FLASH_DRV_Wdog_Status_Clear** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_Wdog_Status_Clear(void) |
| **Function description** | Check flash watchdog timeout status |
| **Input parameter** | Nothing |
| **Output parameter** | Nothing |
| **Return value** | The result of flash watchdog clear status |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### 63.4.20.3 Arguments

None

#### 63.4.20.4 Returns

Please see the definition of status_t.

***Note:*** If the function cannot clear it, you can use any clear function to clear it.

### 63.4.21 FLASH_DRV_EDC_Check

Offset: 50h

#### 63.4.21.1 Function Header

*typedef struct {*
*uint32_t data1;*
*uint32_t data2;*
*uint32_t data3;*
*uint32_t dest;*
*}FLASH_API_ECC_CHECK_CFG_TYPE*

*status_t FLASH_DRV_EDC_Check(FLASH_API_ECC_CHECK_CFG_TYPE * flash_api_cfg)*

#### 63.4.21.2 Description

| **Function name** | **FLASH_DRV_EDC_Check** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_EDC_Check(FLASH_API_ECC_CHECK_CFG_TYPE * flash_api_cfg) |
| **Function description** | Users do self-test on EDC function (Detect the function of EDC by inputting the measured data and expected value. The EDC error flag can be obtained through the read status |

| | |
|---|---|
| | function) |
| **Input parameter** | See section: FLASH_API_ECC_CHECK_CFG_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing EDC check command |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.21.3 Arguments*

FLASH_API_ECC_CHECK_CFG_TYPE is used to pass the parameters required for the flash EDC check.

| **FLASH_API_ECC_CHECK_CFG_TYPE** | **Description** |
|---|---|
| uint32_t data1 | Lower 32 bits of data to be tested |
| uint32_t data2 | 63:32 bits of data to be tested |
| uint32_t data3 | Expected ECC data (8 bits) |
| uint32_t dest | Read address |

#### *63.4.21.4 Returns*

Please see the definition of status_t.

### 63.4.22 FLASH_DRV_ECC_Check

Offset: 54h

#### *63.4.22.1 Function Header*

```
typedef struct {
uint32_t data1;
uint32_t data2;
uint32_t data3;
 uint32_t dest;
}FLASH_API_ECC_CHECK_CFG_TYPE

status_t FLASH_DRV_ECC_Check(FLASH_API_ECC_CHECK_CFG_TYPE * flash_api_cfg)
```

#### *63.4.22.2 Description*

| **Function name** | **FLASH_DRV_ECC_Check** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_ECC_Check(FLASH_API_ECC_CHECK_CFG_TYPE * flash_api_cfg) |
| **Function description** | Users do a self-test on ECC function (Detect the function of ECC by inputting the measured data and expected value, The ECC error flag can be obtained through the read status function) |
| **Input parameter** | See section: FLASH_API_ECC_CHECK_CFG_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing the ECC check command |

| | |
|---|---|
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.22.3 Arguments*

FLASH_API_ECC_CHECK_CFG_TYPE is used to pass the parameters required for the flash EDC check.

| FLASH_API_ECC_CHECK_CFG_TYPE | Description |
|---|---|
| uint32_t data1 | Lower 32 bits of data to be tested |
| uint32_t data2 | 63:32 bits of data to be tested |
| uint32_t data3 | Expected ECC data (8 bits) |
| uint32_t dest | Read address |

#### *63.4.22.4 Returns*

Please see the definition of status_t.

### 63.4.23 FLASH_DRV_Test_Mode_Close

Offset: 58h

#### *63.4.23.1 Function Header*

***status_t FLASH_DRV_Test_Mode_Close (void)***

#### *63.4.23.2 Description*

| | |
|---|---|
| **Function name** | **FLASH_DRV_Test_Mode_Close** |
| **Function prototype** | status_t FLASH_DRV_Test_Mode_Close (void) |
| **Function description** | Close Test mode (after array integrity check, margin read check, ECC/EDC self-check completed) |
| **Input parameter** | Nothing |
| **Output parameter** | Nothing |
| **Return value** | The result of executing the test mode close command |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.23.3 Arguments*

None

#### *63.4.23.4 Returns*

Please see the definition of status_t.

### 63.4.24 FLASH_DRV_Read_Wait_CFG

Offset: 5Ch

#### *63.4.24.1 Function Header*

***typedef struct {***
***FLASH_API_READ_WAIT_LOCK_TYPE lock_sel;***
***FLASH_API_WAIT_STATE_TYPE wait_number;***

*}FLASH_API_READ_WAIT_CFG_TYPE*

*status_t FLASH_DRV_Read_Wait_CFG(FLASH_API_READ_WAIT_CFG_TYPE * flash_api_cfg)*

#### *63.4.24.2 Description*

| | |
|---|---|
| **Function name** | **FLASH_DRV_Read_Wait_CFG** |
| **Function prototype** | status_t FLASH_DRV_Read_Wait_CFG(FLASH_API_READ_WAIT_CFG_TYPE * flash_api_cfg) |
| **Function description** | Configure the number of wait state and lock this value |
| **Input parameter** | See section: FLASH_API_READ_WAIT_CFG_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing read wait config command |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.24.3 Arguments*

FLASH_API_READ_WAIT_CFG_TYPE is used to pass the parameters required for configuring the read wait number and lock.

| **FLASH_API_READ_WAIT_CFG_TYPE** | **Description** |
|---|---|
| FLASH_API_READ_WAIT_LOCK_TYPE lock_sel | Lock wait number configure (Cannot be changed after lock) |
| FLASH_API_WAIT_STATE_TYPE wait_number | Select wait number |

| **FLASH_API_READ_WAIT_LOCK_TYPE** | **Description** |
|---|---|
| FLASH_READ_WAIT_LOCK_DISABLE | Lock disable |
| FLASH_READ_WAIT_LOCK_ENABLE | Lock enable |

| **FLASH_API_WAIT_STATE_TYPE** | **Description** |
|---|---|
| FLASH_WAIT_SELECT1 | Wait for 1 cycle when accessing the flash array |
| FLASH_WAIT_SELECT2 | Wait for 2 cycles when accessing the flash array |
| FLASH_WAIT_SELECT3 | Wait for 3 cycles when accessing the flash array |
| FLASH_WAIT_SELECT4 | Wait for 4 cycles when accessing the flash array |
| FLASH_WAIT_SELECT5 | Wait for 5 cycles when accessing the flash array |
| FLASH_WAIT_SELECT6 | Wait for 6 cycles when accessing the flash array |
| FLASH_WAIT_SELECT7 | Wait for 7 cycles when accessing the flash array |
| FLASH_WAIT_SELECT8 | Wait for 8 cycles when accessing the flash array |
| FLASH_WAIT_HOLD | Keep config unchanged |

#### *63.4.24.4 Returns*

Please see the definition of status_t.

### 63.4.25 FLASH_DRV_ENABLE_HOLD_CFG

Offset: 60h

#### *63.4.25.1 Function Header*

***status_t FLASH_DRV_ENABLE_HOLD_CFG (FLASH_API_AUTO_HOLD_TYPE flash_api_cfg)***

#### *63.4.25.2 Description*

| **Function name** | **FLASH_DRV_ENABLE_HOLD_CFG** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_ENABLE_HOLD_CFG (FLASH_API_AUTO_HOLD_TYPE * flash_api_cfg) |
| **Function description** | Configure flash auto hold function |
| **Input parameter** | See section: FLASH_API_AUTO_HOLD_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing the auto hold command |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.25.3 Arguments*

FLASH_API_AUTO_HOLD_TYPE is used to pass the parameters required for the auto hold function.

| **FLASH_API_AUTO_HOLD_TYPE** | **Description** |
|---|---|
| FLASH_AUTO_HOLD_DISABLE | Disable |
| FLASH_AUTO_HOLD_ENABLE | Enable |

#### *63.4.25.4 Returns*

Please see the definition of status_t.

### 63.4.26 FLASH_DRV_READ_ADDR_FUNC

Offset: 64h

#### *63.4.26.1 Function Header*

***status_t FLASH_DRV_READ_ADDR_FUNC (uint32_t * data0)***

#### *63.4.26.2 Description*

| **Function name** | **FLASH_DRV_READ_ADDR_FUNC** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_READ_ADDR_FUNC (uint32_t * data0) |
| **Function description** | Read the first failing address in related events |
| **Input parameter** | Nothing |
| **Output parameter** | The first failing address |
| **Return value** | The result of executing the FLASH_DRV_READ_ADDR_FUNC command |
| **Precondition** | Nothing |
| **Called function** | Nothing |

Flagchip

#### *63.4.26.3 Arguments*

None

#### *63.4.26.4 Returns*

Please see the definition of status_t.

### 63.4.27 FLASH_DRV_Test_Cfg_FUNC

Offset: 68h

#### *63.4.27.1 Function Header*

```
typedef struct {
FLASH_API_REG_BIT_TYPE ecc_in_cfg;
FLASH_API_REG_BIT_TYPE edc_in_cfg;
FLASH_API_REG_BIT_TYPE ecc_single_bit_cfg;
FLASH_API_REG_BIT_TYPE ae_in_cfg;
FLASH_API_REG_BIT_TYPE ea_in_cfg_hi;
FLASH_API_REG_BIT_TYPE ea_in_cfg_lo;
FLASH_API_REG_BIT_TYPE aibpe_in_cfg;
FLASH_API_REG_BIT_TYPE test128_in_cfg;
FLASH_API_REG_BIT_TYPE edaie_in_cfg_hi;
FLASH_API_REG_BIT_TYPE edaie_in_cfg_lo;
}FLASH_API_TEST_CFG_TYPE

status_t FLASH_DRV_Test_Cfg_FUNC(FLASH_API_TEST_CFG_TYPE * flash_api_cfg)
```

#### *63.4.27.2 Description*

| | |
|---|---|
| **Function name** | *FLASH_DRV_Test_Cfg_FUNC* |
| **Function prototype** | status_t FLASH_DRV_Test_Cfg_FUNC(FLASH_API_TEST_CFG_TYPE * flash_api_cfg) |
| **Function description** | Configure various self-test enable bit |
| **Input parameter** | See section: FLASH_API_TEST_CFG_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.27.3 Arguments*

FLASH_API_TEST_CFG_TYPE is used to pass the parameters required for the auto hold function.

| **FLASH_API_TEST_CFG_TYPE** | **Description** |
|---|---|
| FLASH_API_REG_BIT_TYPE ecc_in_cfg; | Self-test double-bit check control |
| FLASH_API_REG_BIT_TYPE edc_in_cfg; | Self-test single-bit check control |
| FLASH_API_REG_BIT_TYPE ecc_single_bit_cfg; | Self-test EDC check control |
| FLASH_API_REG_BIT_TYPE ae_in_cfg; | Self-test address ECC check control |
| FLASH_API_REG_BIT_TYPE ea_in_cfg_hi; | Self-test EA hi check control |

| | |
|---|---|
| FLASH_API_REG_BIT_TYPE ea_in_cfg_lo; | Self-test EA lo check control |
| FLASH_API_REG_BIT_TYPE aibpe_in_cfg; | Array integrity breakpoint control |
| FLASH_API_REG_BIT_TYPE test128_in_cfg; | Self-test parallel mode check control |
| FLASH_API_REG_BIT_TYPE edaie_in_cfg_hi; | Self-test edc hi address injection check control |
| FLASH_API_REG_BIT_TYPE edaie_in_cfg_lo; | Self-test edc lo address injection check control |

| **FLASH_API_REG_BIT_TYPE** | **Description** |
|---|---|
| FLASH_REG_BIT_CFG_DISABLE | Disable reg bit |
| FLASH_REG_BIT_CFG_ENABLE | Enable reg bit |
| FLASH_REG_BIT_CFG_HOLD | Hold reg bit |

#### *63.4.27.4 Returns*

Please see the definition of status_t.

### 63.4.28 FLASH_DRV_Array_Integrity_Suspend_FUNC

Offset: 6Ch

#### *63.4.28.1 Function Header*

***status_t FLASH_DRV_Array_Integrity_Suspend_FUNC(FLASH_API_REG_BIT_TYPE flash_api_cfg)***

#### *63.4.28.2 Description*

| | |
|---|---|
| **Function name** | ***FLASH_DRV_Array_Integrity_Suspend_FUNC*** |
| **Function prototype** | status_t FLASH_DRV_Array_Integrity_Suspend_FUNC(FLASH_API_REG_BIT_TYPE * flash_api_cfg) |
| **Function description** | Configure array integrity suspend function |
| **Input parameter** | See section: FLASH_API_REG_BIT_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.28.3 Arguments*

| **FLASH_API_REG_BIT_TYPE** | **Description** |
|---|---|
| FLASH_REG_BIT_CFG_DISABLE | Disable reg bit |
| FLASH_REG_BIT_CFG_ENABLE | Enable reg bit |
| FLASH_REG_BIT_CFG_HOLD | Hold reg bit |

#### *63.4.28.4 Returns*

Please see the definition of status_t.

### 63.4.29 FLASH_DRV_Test_Mode_Open

Offset: 70h

#### *63.4.29.1 Function Header*

***status_t FLASH_DRV_Test_Mode_Open (void)***

#### *63.4.29.2 Description*

| | |
|---|---|
| **Function name** | **FLASH_DRV_Test_Mode_Open** |
| **Function prototype** | status_t FLASH_DRV_Test_Mode_Open (void) |
| **Function description** | Open Test mode (before array integrity check, margin read check, ECC/EDC self-check) |
| **Input parameter** | Nothing |
| **Output parameter** | Nothing |
| **Return value** | The result of executing the test mode copen command |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.29.3 Arguments*

None

#### *63.4.29.4 Returns*

Please see the definition of status_t.

### 63.4.30 FLASH_DRV_Sector_Program

Offset: 74h

#### *63.4.30.1 Function Header*

***status_t FLASH_DRV_Sector_Program (uint32_t *dest, uint32_t *addr, uint32_t *info)***

#### *63.4.30.2 Description*

| | |
|---|---|
| **Function name** | FLASH_DRV_Sector_Program |
| **Function prototype** | status_t FLASH_DRV_Sector_Program (uint32_t *dest, uint32_t *addr, uint32_t *info) |
| **Function description** | Execute flash sector program operation |
| **Input parameter** | dest: program flash address header<br>addr: need to be programmed data address (typical ram address)<br>info: function execute flag |
| **Output parameter** | Nothing |
| **Return value** | The result of executing the flash_sector program |
| **Precondition** | Unlock flash |
| **Called function** | Nothing |

#### *63.4.30.3 Arguments*

N/A

#### 63.4.30.4 Returns

Please see the definition of status_t.

### 63.4.31 FLASH_DRV_Read_Status_Check_128

Offset : 78h

#### 63.4.31.1 Function Header

*uint32_t FLASH_DRV_Read_Status_Check_128(FLASH_API_READ_STATUS_CHECK_TYPE flash_api_cfg)*

#### 63.4.31.2 Description

| **Function name** | **FLASH_DRV_Read_Status_Check_128** |
|---|---|
| **Function prototype** | uint32_t FLASH_DRV_Read_Status_Check_128(FLASH_API_READ_STATUS_CHECK_TYPE flash_api_cfg) |
| **Function description** | Check the status of pflash in parallel mode |
| **Input parameter** | See section: FLASH_API_READ_STATUS_CHECK_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of the flash status |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### 63.4.31.3 Arguments

FLASH_API_READ_STATUS_CHECK_TYPE is used to pass the parameters.

| **FLASH_API_READ_STATUS_CHECK_TYPE** | **Description** |
|---|---|
| FLASH_ECC_CHECK_MASK | Check double bit detection occurred |
| FLASH_SBC_CHECK_MASK | Check single bit correction occurred |
| FLASH_AEE_CHECK_MASK | Check read-while-write error occurred |
| FLASH_EDC_CHECK_MASK | Check EDC after an ECC error occurred |
| FLASH_SPACE_CHECK_MASK | Check Space status |
| FLASH_TWO_CHECK_MASK | Check two status |
| FLASH_ONE_CHECK_MASK | Check one status |
| FLASH_BLANK_CHECK_MASK | Check Blank status |
| STATUS_LC_SCAN_CHECK_MASK | Check Flash scan error |
| FLASH_HSM_LD_CHECK_MASK | Check HSM load error |
| FLASH_NVR_KEY_CHECK_MASK | Check NVR Key error |

#### 63.4.31.4 Returns

Please see the definition of status_t.

### 63.4.32 FLASH_DRV_Read_Status_Check_Clear_128

Offset: 7Ch

#### 63.4.32.1 Function Header

*status_t FLASH_DRV_Read_Status_Check_Clear_128(FLASH_API_READ_STATUS_CLEAR_TYPE flash_api_cfg)*

#### 63.4.32.2 Description

| | |
|---|---|
| **Function name** | **FLASH_DRV_Read_Status_Check_Clear_128** |
| **Function prototype** | status_t FLASH_DRV_Read_Status_Check_Clear_128 (FLASH_API_READ_STATUS_CLEAR_TYPE flash_api_cfg) |
| **Function description** | Clear pflash(block2/3) status flag |
| **Input parameter** | See section: FLASH_API_READ_STATUS_CLEAR_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### 63.4.32.3 Arguments

FLASH_API_READ_STATUS_CLEAR_TYPE is used to pass the parameters.

| FLASH_API_READ_STATUS_CLEAR_TYPE | Description |
|---|---|
| FLASH_CLEAR_ECC_MASK | Clear double-bit error |
| FLASH_CLEAR_SBC_MASK | Clear single-bit error |
| FLASH_CLEAR_AEE_MASK | Clear AEE error |
| FLASH_CLEAR_EDC_MASK | Clear EDC error |
| FLASH_CLEAR_SPACE__MASK | Clear Space error |
| FLASH_CLEAR_TWO_MASK | Clear Two error |
| FLASH_CLEAR_One_MASK | Clear One error |
| FLASH_CLEAR_BLANK_MASK | Clear Blank error |

#### 63.4.32.4 Returns

Please see the definition of status_t.

### 63.4.33 FLASH_DRV_Array_Integrity_Check_128

Offset: 80h

#### 63.4.33.1 Function Header

```
typedef struct {
uint32_t dest;
uint32_t sed0;
uint32_t sed1;
uint32_t sed2;
uint32_t sed3;
uint32_t sed4;
uint32_t sed5;
```

*FLASH_API_WAIT_STATE_TYPE wait_number;*

*FLASH_API_MARGIN_LEVEL_TYPE mrv;*

*FLASH_API_ARRAY_INTEGRITY_SEQ_TYPE als;*

*FLASH_API_REG_BIT_TYPE bpe;*

*}FLASH_API_TEST_CHECK_CFG_TYPE;*

*status_t FLASH_DRV_Array_Integrity_Check_128(FLASH_API_TEST_CHECK_CFG_TYPE * flash_api_cfg)*

#### *63.4.33.2 Description*

| **Function name** | **FLASH_DRV_Array_Integrity_Check_128** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_Array_Integrity_Check_128(FLASH_API_TEST_CHECK_CFG_TYPE * flash_api_cfg) |
| **Function description** | Check flash array |
| **Input parameter** | See section: FLASH_API_TEST_CHECK_CFG_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing the array integrity command |
| **Precondition** | FLASH TEST MODE OPEN |
| **Called function** | Nothing |

#### *63.4.33.3 Arguments*

FLASH_API_TEST_CHECK_CFG_TYPE is used to pass the parameters required for the flash array integrity check.

| **FLASH_API_TEST_CHECK_CFG_TYPE** | **Description** |
|---|---|
| uint32_t dest | Array integrity check address |
| uint32_t sed0 | Seed value |
| uint32_t sed1 | Seed value |
| uint32_t sed2 | Seed value |
| uint32_t sed3 | Seed value |
| uint32_t sed4 | Seed value |
| uint32_t sed5 | Seed value |
| FLASH_API_WAIT_STATE_TYPE wait_number | The number of wait states |
| FLASH_API_MARGIN_LEVEL_TYPE mrv | Margin read value (not required) |
| FLASH_ARRAY_INTEGRITY_SEQ_TYPE als | Select the sequence of array integrity check |
| FLASH_API_REG_BIT_TYPE bpe | Enable breakpoint function |

| **FLASH_API_WAIT_STATE_TYPE** | **Description** |
|---|---|
| FLASH_WAIT_SELECT1 | Wait for 1 cycle when accessing the flash array |
| FLASH_WAIT_SELECT2 | Wait for 2 cycles when accessing the flash array |
| FLASH_WAIT_SELECT3 | Wait for 3 cycles when accessing the flash array |
| FLASH_WAIT_SELECT4 | Wait for 4 cycles when accessing the flash array |
| FLASH_WAIT_SELECT5 | Wait for 5 cycles when accessing the flash array |
| FLASH_WAIT_SELECT6 | Wait for 6 cycles when accessing the flash array |

| | |
|---|---|
| FLASH_WAIT_SELECT7 | Wait for 7 cycles when accessing the flash array |
| FLASH_WAIT_SELECT8 | Wait for 8 cycles when accessing the flash array |
| FLASH_WAIT_HOLD | Keep config unchanged |

| **FLASH_API_MARGIN_LEVEL_TYPE** | **Description** |
|---|---|
| FLASH_MARGIN_LEVEL0 | Margin level 0 (programmed level) |
| FLASH_MARGIN_LEVEL1 | Margin level 1 (erased level) |

| **FLASH_API_ARRAY_SEQ_TYPE** | **Description** |
|---|---|
| FLASH_ARRAY_SEQ1 | Select proprietary sequence |
| FLASH_ARRAY_SEQ2 | Select sequential sequence |

| **FLASH_API_REG_BIT_TYPE** | **Description** |
|---|---|
| FLASH_REG_BIT_CFG_DISABLE | Disable reg bit |
| FLASH_REG_BIT_CFG_ENABLE | Enable reg bit |
| FLASH_REG_BIT_CFG_HOLD | Hold reg bit |

#### *63.4.33.4 Returns*

Please see the definition of status_t.

### 63.4.34 FLASH_DRV_Margin_Read_Check_128

Offset: 84h

#### *63.4.34.1 Function Header*

```
typedef struct {
uint32_t dest;
uint32_t sed0;
uint32_t sed1;
uint32_t sed2;
uint32_t sed3;
uint32_t sed4;
uint32_t sed5;
FLASH_API_WAIT_STATE_TYPE wait_number;
FLASH_API_MARGIN_LEVEL_TYPE mrv;
FLASH_API_ARRAY_INTEGRITY_SEQ_TYPE als;
FLASH_API_REG_BIT_TYPE bpe;
}FLASH_API_TEST_CHECK_CFG_TYPE;

status_t FLASH_DRV_Margin_Read_Check_128(FLASH_API_TEST_CHECK_CFG_TYPE *  flash_api_cfg)
```

#### *63.4.34.2 Description*

| **Function name** | **FLASH_DRV_Margin_Read_Check_128** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_Margin_Read_Check_128(FLASH_API_TEST_CHECK_CFG_TYPE * |

| | |
|---|---|
| | flash_api_cfg) |
| **Function description** | Check margin read |
| **Input parameter** | See section: FLASH_API_TEST_CHECK_CFG_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing the margin read command |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.34.3 Arguments*

FLASH_API_TEST_CHECK_CFG_TYPE is used to pass the parameters required for the flash margin read check.

| **FLASH_API_TEST_CHECK_CFG_TYPE** | **Description** |
|---|---|
| uint32_t dest | Array integrity check address |
| uint32_t sed0 | Seed value |
| uint32_t sed1 | Seed value |
| uint32_t sed2 | Seed value |
| uint32_t sed3 | Seed value |
| uint32_t sed4 | Seed value |
| uint32_t sed5 | Seed value |
| FLASH_API_WAIT_STATE_TYPE wait_number | The number of wait states |
| FLASH_API_MARGIN_LEVEL_TYPE mrv | Margin read value (not required) |
| FLASH_ARRAY_INTEGRITY_SEQ_TYPE als | Select the sequence of array integrity check |
| FLASH_API_REG_BIT_TYPE bpe | Enable breakpoint function |

| **FLASH_API_WAIT_STATE_TYPE** | **Description** |
|---|---|
| FLASH_WAIT_SELECT1 | Wait for 1 cycle when accessing the flash array |
| FLASH_WAIT_SELECT2 | Wait for 2 cycles when accessing the flash array |
| FLASH_WAIT_SELECT3 | Wait for 3 cycles when accessing the flash array |
| FLASH_WAIT_SELECT4 | Wait for 4 cycles when accessing the flash array |
| FLASH_WAIT_SELECT5 | Wait for 5 cycles when accessing the flash array |
| FLASH_WAIT_SELECT6 | Wait for 6 cycles when accessing the flash array |
| FLASH_WAIT_SELECT7 | Wait for 7 cycles when accessing the flash array |
| FLASH_WAIT_SELECT8 | Wait for 8 cycles when accessing the flash array |
| FLASH_WAIT_HOLD | Keep config unchanged |

| **FLASH_API_MARGIN_LEVEL_TYPE** | **Description** |
|---|---|
| FLASH_MARGIN_LEVEL0 | Margin level 0 (programmed level) |
| FLASH_MARGIN_LEVEL1 | Margin level 1 (erased level) |

| **FLASH_API_ARRAY_SEQ_TYPE** | **Description** |
|---|---|
| FLASH_ARRAY_SEQ1 | Select proprietary sequence |

| | |
|---|---|
| FLASH_ARRAY_SEQ2 | Select sequential sequence |

| FLASH_API_REG_BIT_TYPE | Description |
|---|---|
| FLASH_REG_BIT_CFG_DISABLE | Disable reg bit |
| FLASH_REG_BIT_CFG_ENABLE | Enable reg bit |
| FLASH_REG_BIT_CFG_HOLD | Hold reg bit |

#### *63.4.34.4 Returns*

Please see the definition of status_t.

### 63.4.35 FLASH_DRV_CTL_CFG_FUNC

Offset: 88h

#### *63.4.35.1 Function Header*

```
typedef struct {
FLASH_API_REG_BIT_TYPE safer;
FLASH_API_REG_BIT_TYPE pgff;
FLASH_API_REG_BIT_TYPE ifr_err_clr;
FLASH_API_REG_BIT_TYPE stop_ack_en;
FLASH_API_REG_BIT_TYPE fmc_cg;
FLASH_API_REG_BIT_TYPE bist_cg;
FLASH_API_REG_BIT_TYPE dpd;
FLASH_API_REG_BIT_TYPE dpd_stop_en;
}FLASH_API_CTL_CFG_TYPE

status_t FLASH_DRV_CTL_CFG_FUNC(FLASH_API_CTL_CFG_TYPE * flash_api_cfg)
```

#### *63.4.35.2 Description*

| | |
|---|---|
| **Function name** | **FLASH_DRV_CTL_CFG_FUNC** |
| **Function prototype** | status_t FLASH_DRV_CTL_CFG_FUNC(FLASH_API_CTL_CFG_TYPE *flash_api_cfg) |
| **Function description** | Configure some flash function ctrl |
| **Input parameter** | See section: FLASH_API_CTL_CFG_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.35.3 Arguments*

FLASH_API_CTL_CFG_TYPE is used to pass the parameters.

| FLASH_API_CTL_CFG_TYPE | Description |
|---|---|
| FLASH_API_REG_BIT_TYPE safer; | Control other errors whether to create response error |
| FLASH_API_REG_BIT_TYPE pgff; | Program full FF into flash array |
| FLASH_API_REG_BIT_TYPE ifr_err_clr; | nvr_ecc_clr control bit |

| | |
|---|---|
| FLASH_API_REG_BIT_TYPE stop_ack_en; | Stop mode ack control bit |
| FLASH_API_REG_BIT_TYPE fmc_cg; | FMC controller clock gating control bit |
| FLASH_API_REG_BIT_TYPE bist_cg; | BIST clock gating control bit |
| FLASH_API_REG_BIT_TYPE dpd; | dpd mode control |
| FLASH_API_REG_BIT_TYPE dpd_stop_en; | dpd stop enable control |

| **FLASH_API_REG_BIT_TYPE** | **Description** |
|---|---|
| FLASH_REG_BIT_CFG_DISABLE | Disable reg bit |
| FLASH_REG_BIT_CFG_ENABLE | Enable reg bit |
| FLASH_REG_BIT_CFG_HOLD | Hold reg bit |

#### *63.4.35.4 Returns*

Please see the definition of status_t.

### 63.4.36 FLASH_DRV_READ_MASTER_ID_FUNC

Offset: 8Ch

#### *63.4.36.1 Function Header*

***status_t FLASH_DRV_READ_MASTER_ID_FUNC(uint8_t * master_id)***

#### *63.4.36.2 Description*

| **Function name** | **FLASH_DRV_READ_MASTER_ID_FUNC** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_READ_MASTER_ID_FUNC(uint8_t *master_id) |
| **Function description** | Acquire master id of previous read pflash(block0/1) error occurred |
| **Input parameter** | Nothing |
| **Output parameter** | Master id |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.36.3 Arguments*

Nothing

#### *63.4.36.4 Returns*

Please see the definition of status_t.

### 63.4.37 FLASH_DRV_EDC_Check_128

Offset: 90h

#### *63.4.37.1 Function Header*

***typedef struct {***
***uint32_t data1;***
***uint32_t data2;***

*uint32_t data3;*

*uint32_t dest;*

*}FLASH_API_ECC_CHECK_CFG_TYPE*

*status_t FLASH_DRV_EDC_Check_128(FLASH_API_ECC_CHECK_CFG_TYPE * flash_api_cfg)*

#### *63.4.37.2 Description*

| **Function name** | **FLASH_DRV_EDC_Check_128** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_EDC_Check_128(FLASH_API_ECC_CHECK_CFG_TYPE * flash_api_cfg) |
| **Function description** | Users do self-test on EDC function (Detect the function of EDC by inputting the measured data and expected value. The EDC error flag can be obtained through the read status function) |
| **Input parameter** | See section: FLASH_API_ECC_CHECK_CFG_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing EDC check command |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.37.3 Arguments*

FLASH_API_ECC_CHECK_CFG_TYPE is used to pass the parameters required for the flash EDC check.

| **FLASH_API_ECC_CHECK_CFG_TYPE** | **Description** |
|---|---|
| uint32_t data1 | Lower 32 bits of data to be tested |
| uint32_t data2 | 63:32 bits of data to be tested |
| uint32_t data3 | Expected ECC data (8 bits) |
| uint32_t dest | Read address |

#### *63.4.37.4 Returns*

Please see the definition of status_t.

### 63.4.38 FLASH_DRV_ECC_Check_128

Offset: 94h

#### *63.4.38.1 Function Header*

*typedef struct {*

*uint32_t data1;*

*uint32_t data2;*

*uint32_t data3;*

*uint32_t dest;*

*}FLASH_API_ECC_CHECK_CFG_TYPE*

*status_t FLASH_DRV_ECC_Check_128(FLASH_API_ECC_CHECK_CFG_TYPE * flash_api_cfg)*

#### 63.4.38.2 Description

| Function name | FLASH_DRV_ECC_Check_128 |
|---|---|
| Function prototype | status_t FLASH_DRV_ECC_Check_128(FLASH_API_ECC_CHECK_CFG_TYPE * flash_api_cfg) |
| Function description | Users do a self-test on ECC function (Detect the function of ECC by inputting the measured data and expected value, The ECC error flag can be obtained through the read status function) |
| Input parameter | See section: FLASH_API_ECC_CHECK_CFG_TYPE to view more allowable value ranges of this parameter |
| Output parameter | Nothing |
| Return value | The result of executing the ECC check command |
| Precondition | Nothing |
| Called function | Nothing |

#### 63.4.38.3 Arguments

FLASH_API_ECC_CHECK_CFG_TYPE is used to pass the parameters required for the flash EDC check.

| FLASH_API_ECC_CHECK_CFG_TYPE | Description |
|---|---|
| uint32_t data1 | Lower 32 bits of data to be tested |
| uint32_t data2 | 63:32 bits of data to be tested |
| uint32_t data3 | Expected ECC data (8 bits) |
| uint32_t dest | Read address |

#### 63.4.38.4 Returns

Please see the definition of status_t.

### 63.4.39 FLASH_DRV_EAC_CHECK_128_HI

Offset: 98h

#### 63.4.39.1 Function Header

*status_t FLASH_DRV_EAC_CHECK_128_HI(uint32_t addr)*

#### 63.4.39.2 Description

| Function name | FLASH_DRV_EAC_CHECK_128_HI |
|---|---|
| Function prototype | status_t FLASH_DRV_EAC_CHECK_128_HI(uint32_t addr) |
| Function description | Users do self-test on EAC Hi function, write into destination address in parallel mode |
| Input parameter | addr: access flash array address |
| Output parameter | Nothing |
| Return value | The result of executing this function |
| Precondition | Nothing |
| Called function | Nothing |

#### 63.4.39.3 Arguments

Nothing

#### 63.4.39.4 Returns

Please see the definition of status_t.

### 63.4.40 FLASH_DRV_AEC_CHECK

Offset: 9Ch

#### 63.4.40.1 Function Header

*status_t FLASH_DRV_AEC_CHECK(uint32_t addr)*

#### 63.4.40.2 Description

| | |
|---|---|
| **Function name** | **FLASH_DRV_AEC_CHECK** |
| **Function prototype** | status_t FLASH_DRV_AEC_CHECK(uint32_t addr) |
| **Function description** | Users do self-test on AEC function, write into destination address |
| **Input parameter** | addr: access flash array address |
| **Output parameter** | Nothing |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### 63.4.40.3 Arguments

Nothing

#### 63.4.40.4 Returns

Please see the definition of status_t

### 63.4.41 FLASH_DRV_MCTL_CFG_FUNC

Offset: A0h

#### 63.4.41.1 Function Header

*typedef struct {*
*FLASH_API_REG_BIT_TYPE ehv;*
*FLASH_API_REG_BIT_TYPE ers;*
*FLASH_API_REG_BIT_TYPE ess;*
*FLASH_API_REG_BIT_TYPE pgm;*
*FLASH_API_REG_BIT_TYPE wdie;*
*FLASH_API_REG_BIT_TYPE aeeie;*
*FLASH_API_REG_BIT_TYPE blankie;*
*FLASH_API_REG_BIT_TYPE pecie;*
*FLASH_API_REG_BIT_TYPE encie;*
*FLASH_API_REG_BIT_TYPE edcie;*
*FLASH_API_REG_BIT_TYPE blankied;*

*FLASH_API_REG_BIT_TYPE blankerr;*

*FLASH_API_REG_BIT_TYPE blankerrd;*

*}FLASH_API_MCTL_CFG_TYPE*

*status_t FLASH_DRV_MCTL_CFG_FUNC(FLASH_API_MCTL_CFG_TYPE * flash_api_cfg)*

### *63.4.41.2 Description*

| **Function name** | **FLASH_DRV_MCTL_CFG_FUNC** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_MCTL_CFG_FUNC(FLASH_API_MCTL_CFG_TYPE *flash_api_cfg) |
| **Function description** | Configure some flash function (program or erase) by main interface |
| **Input parameter** | See section: FLASH_API_MCTL_CFG_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

### *63.4.41.3 Arguments*

FLASH_API_MCTL_CFG_TYPE is used to pass the parameters.

| **FLASH_API_MCTL_CFG_TYPE** | **Description** |
|---|---|
| FLASH_API_REG_BIT_TYPE ehv*;* | High voltage control bit |
| FLASH_API_REG_BIT_TYPE ers*;* | Erase sequence control bit |
| FLASH_API_REG_BIT_TYPE ess*;* | Erase type select |
| FLASH_API_REG_BIT_TYPE pgm*;* | Program sequence control bit |
| FLASH_API_REG_BIT_TYPE wdie*;* | Flash watchdog control bit |
| FLASH_API_REG_BIT_TYPE aeeie*;* | Flash AEE control bit |
| FLASH_API_REG_BIT_TYPE blankie*;* | Flash blank control bit |
| FLASH_API_REG_BIT_TYPE pecie*;* | Flash program or erase complete interrupt control bit |
| FLASH_API_REG_BIT_TYPE encie*;* | Flash enc control bit |
| FLASH_API_REG_BIT_TYPE edcie*;* | Flash edc control bit |
| FLASH_API_REG_BIT_TYPE blankied*;* | Data Flash blank control bit |
| FLASH_API_REG_BIT_TYPE blankerr*;* | Flash blank error control bit |
| FLASH_API_REG_BIT_TYPE blankerrd*;* | Data flash blank error control bit |

| **FLASH_API_REG_BIT_TYPE** | **Description** |
|---|---|
| FLASH_REG_BIT_CFG_DISABLE | Disable reg bit |
| FLASH_REG_BIT_CFG_ENABLE | Enable reg bit |
| FLASH_REG_BIT_CFG_HOLD | Hold reg bit |

### *63.4.41.4 Returns*

Please see the definition of status_t.

### 63.4.42 FLASH_DRV_AMCTL_CFG_FUNC

Offset: A4h

#### *63.4.42.1 Function Header*

*typedef struct {*
*FLASH_API_REG_BIT_TYPE aehv;*
*FLASH_API_REG_BIT_TYPE aers;*
*FLASH_API_REG_BIT_TYPE aess;*
*FLASH_API_REG_BIT_TYPE apgm;*
*FLASH_API_REG_BIT_TYPE awdie;*
*FLASH_API_REG_BIT_TYPE apecie;*
*}FLASH_API_AMCTL_CFG_TYPE*

*status_t FLASH_DRV_AMCTL_CFG_FUNC(FLASH_API_AMCTL_CFG_TYPE * flash_api_cfg)*

#### *63.4.42.2 Description*

| **Function name** | **FLASH_DRV_AMCTL_CFG_FUNC** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_AMCTL_CFG_FUNC(FLASH_API_AMCTL_CFG_TYPE *flash_api_cfg) |
| **Function description** | Configure some flash function (program or erase) by alt interface |
| **Input parameter** | See section: FLASH_API_AMCTL_CFG_TYPE to view more allowable value ranges of this parameter |
| **Output parameter** | Nothing |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.42.3 Arguments*

FLASH_API_MCTL_CFG_TYPE is used to pass the parameters.

| **FLASH_API_MCTL_CFG_TYPE** | **Description** |
|---|---|
| FLASH_API_REG_BIT_TYPE aehv*;* | High voltage control bit |
| FLASH_API_REG_BIT_TYPE aers*;* | Erase sequence control bit |
| FLASH_API_REG_BIT_TYPE aess*;* | Erase type select |
| FLASH_API_REG_BIT_TYPE apgm*;* | Program sequence control bit |
| FLASH_API_REG_BIT_TYPE awdie*;* | Flash watchdog control bit |
| FLASH_API_REG_BIT_TYPE apecie*;* | Flash program or erase complete interrupt control bit |

| **FLASH_API_REG_BIT_TYPE** | **Description** |
|---|---|
| FLASH_REG_BIT_CFG_DISABLE | Disable reg bit |
| FLASH_REG_BIT_CFG_ENABLE | Enable reg bit |
| FLASH_REG_BIT_CFG_HOLD | Hold reg bit |

#### 63.4.42.4 Returns

Please see the definition of status_t.

### 63.4.43 FLASH_DRV_READ_PE_MASTER_ID_FUNC

Offset: A8h

#### 63.4.43.1 Function Header

*status_t FLASH_DRV_READ_PE_MASTER_ID_FUNC(uint8_t * master_id)*

#### 63.4.43.2 Description

| **Function name** | **FLASH_DRV_READ_PE_MASTER_ID_FUNC** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_READ_PE_MASTER_ID_FUNC(uint8_t *master_id) |
| **Function description** | Acquire master id of operate flash array currently by main interface |
| **Input parameter** | Nothing |
| **Output parameter** | Master id |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### 63.4.43.3 Arguments

Nothing

#### 63.4.43.4 Returns

Please see the definition of status_t.

### 63.4.44 FLASH_DRV_READ_PE_AMASTER_ID_FUNC

Offset: ACh

#### 63.4.44.1 Function Header

*status_t FLASH_DRV_READ_PE_AMASTER_ID_FUNC(uint8_t * master_id)*

#### 63.4.44.2 Description

| **Function name** | **FLASH_DRV_READ_PE_AMASTER_ID_FUNC** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_READ_PE_AMASTER_ID_FUNC(uint8_t *master_id) |
| **Function description** | Acquire master ID of operate flash array currently by alt interface |
| **Input parameter** | Nothing |
| **Output parameter** | Master id |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### 63.4.44.3 Arguments

Nothing

#### 63.4.44.4 Returns

Please see the definition of status_t.

### 63.4.45 FLASH_DRV_EAC_CHECK_128_LO

Offset: B0h

#### *63.4.45.1 Function Header*

***status_t FLASH_DRV_EAC_CHECK_128_LO(uint32_t addr)***

#### *63.4.45.2 Description*

| **Function name** | **FLASH_DRV_EAC_CHECK_128_LO** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_EAC_CHECK_128_LO(uint32_t addr) |
| **Function description** | Users do self-test on EAC LO function, write into destination address in parallel mode |
| **Input parameter** | addr: access flash array address |
| **Output parameter** | Nothing |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### *63.4.45.3 Arguments*

Nothing

#### *63.4.45.4 Returns*

Please see the definition of status_t.

### 63.4.46 FLASH_DRV_EDCA_CHECK_128_Hi

Offset: B4h

#### *63.4.46.1 Function Header*

***status_t FLASH_DRV_EDCA_CHECK_128_HI(uint32_t addr)***

#### *63.4.46.2 Description*

| **Function name** | **FLASH_DRV_EDCA_CHECK_128_HI** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_EDCA_CHECK_128_HI(uint32_t addr) |
| **Function description** | Users do self-test on EDCA HI function, write into destination address in parallel mode |
| **Input parameter** | addr: access flash array address |
| **Output parameter** | Nothing |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### 63.4.46.3 Arguments

Nothing

#### 63.4.46.4 Returns

Please see the definition of status_t.

### 63.4.47 FLASH_DRV_EDCA_CHECK_128_LO

Offset: B8h

#### 63.4.47.1 Function Header

*status_t FLASH_DRV_EDCA_CHECK_128_LO(uint32_t addr)*

#### 63.4.47.2 Description

| **Function name** | **FLASH_DRV_EDCA_CHECK_128_LO** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_EDCA_CHECK_128_LO(uint32_t addr) |
| **Function description** | Users do self-test on EDCA LO function, write into destination address in parallel mode |
| **Input parameter** | addr: access flash array address |
| **Output parameter** | Nothing |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### 63.4.47.3 Arguments

Nothing

#### 63.4.47.4 Returns

Please see the definition of status_t.

### 63.4.48 FLASH_DRV_ABORT_PE

Offset: BCh

#### 63.4.48.1 Function Header

*status_t FLASH_DRV_ABORT_PE(void)*

#### 63.4.48.2 Description

| **Function name** | **FLASH_DRV_ABORT_PE** |
|---|---|
| **Function prototype** | status_t FLASH_DRV_ABORT_PE(void) |
| **Function description** | Users to abort flash program/erase operation |
| **Input parameter** | Nothing |
| **Output parameter** | Nothing |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

Flagchip

#### 63.4.48.3 Arguments

Nothing

#### 63.4.48.4 Returns

Please see the definition of status_t.

### 63.4.49 FLASH_DRV_WRITE_ADR_FUNC

Offset: C0h

#### 63.4.49.1 Function Header

*status_t FLASH_DRV_WRITE_ADR_FUNC(uint32_t data0)*

#### 63.4.49.2 Description

| | |
|---|---|
| **Function name** | **FLASH_DRV_WRITE_ADR_FUNC** |
| **Function prototype** | status_t FLASH_DRV_WRITE_ADR_FUNC(uint32_t data0) |
| **Function description** | Users to a address to adr register |
| **Input parameter** | data0: write adr register data |
| **Output parameter** | Nothing |
| **Return value** | The result of executing this function |
| **Precondition** | Nothing |
| **Called function** | Nothing |

#### 63.4.49.3 Arguments

Nothing

#### 63.4.49.4 Returns

Please see the definition of status_t.

Flagchip